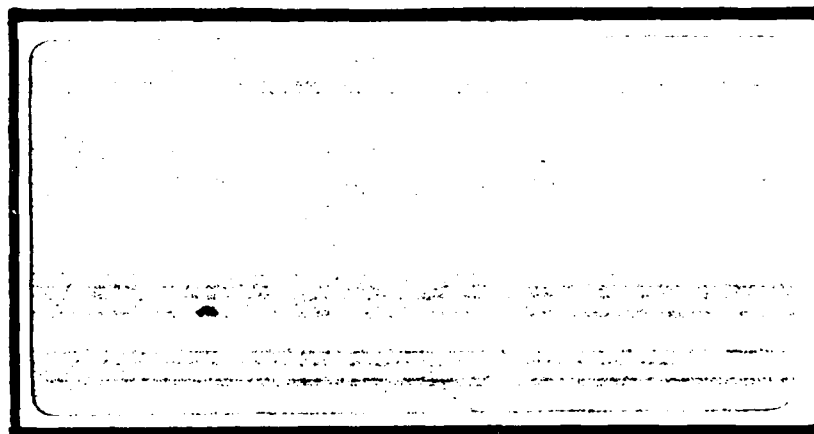


DTIC

FILE

COPY

AD-A216 376



DTIC

ELECTE

JAN 03 1990

DCS

**DISTRIBUTION STATEMENT A**

Approved for public release  
Distribution Unlimited

DEPARTMENT OF THE AIR FORCE

AIR UNIVERSITY

**AIR FORCE INSTITUTE OF TECHNOLOGY**

Wright-Patterson Air Force Base, Ohio

90 01 02 081

AFIT/GEP/ENP/89D-5

DTIC  
S ELECTE  
JAN 03 1993  
D

A Numerical Solution to the Boltzmann Equation  
for Use in Calculating Pumping Rates  
in a CO<sub>2</sub> Discharge Laser

THESIS

David Alan Honey  
Captain, USAF

AFIT/GEP/ENP/89D-5

Approved for public release; distribution unlimited.

A Numerical Solution to the Boltzmann Equation  
for Use in Calculating Pumping Rates  
in a CO2 Discharge Laser

THESIS

Presented to the Faculty of the School of Engineering  
of the Air Force Institute of Technology

Air University

In Partial Fulfillment of the  
Requirements for the Degree of  
Master of Science in Engineering Physics

David Alan Honey, B.S., M.S.  
Captain, USAF

December, 1989

Approved for public release; distribution unlimited.



Accession For	
NTIS	CRA&I <input checked="checked" type="checkbox"/>
DTIC	TAB <input type="checkbox"/>
Unannounced	<input type="checkbox"/>
Justification	
By	
Distribution/	
Availability Codes	
Dist	Avail and/or Special
A-1	

## *Preface*

The purpose of this study was to modify a CO<sub>2</sub> laser design program so that it would calculate pumping terms through numerical solution of the collisional Boltzmann equation. Both the laser design code and the Boltzmann solver routine were written in Microsoft QuickBasic for use on an IBM PC or equivalent. Transport coefficients were computed from the electron number density distributions output by the Boltzmann routine.

Four methods of solving the Boltzmann equation were tried, with L-U decomposition giving the most accurate results and the shortest run times. Good agreement was found between the predicted transport coefficients and the results reported by others. Areas where the program had difficulty were documented and should be more thoroughly investigated. A few of the effects on laser output of Boltzmann related parameters were also studied.

In performing this project I had a great deal of help and guidance from others. I wish to thank my thesis advisor, Maj Stone, for starting this project and for helping me solve some very tough problems. I also wish to thank Dr Bailey for the many discussions we had and the insights he provided. Finally, I wish to thank my wife Cynthia for all her support and understanding during this project.

David Alan Honey

## *Table of Contents*

	Page
Preface . . . . .	ii
Table of Contents . . . . .	iii
Abstract . . . . .	vi
 I. The CO2 Laser Model . . . . .	 1
 II. The Boltzmann Equation . . . . .	 3
2.1 Derivation of the Boltzmann Equation and Important Assumptions . . . .	3
2.2 Energy Transfer by Elastic Collisions, the Applied Electric Fields, and In- elastic Collisions . . . . .	5
2.2.1 Elastic Collisions . . . . .	5
2.2.2 Applied Electric Field . . . . .	6
2.2.3 Inelastic Collisions . . . . .	8
2.3 Finite Difference Form of the Boltzmann Equation . . . . .	10
2.4 Numerical Solution of the Boltzmann Equation . . . . .	13
2.5 Transport Coefficients . . . . .	14
2.5.1 Drift Velocity . . . . .	15
2.5.2 Rate of Excitation . . . . .	15
 III. Computer Programs . . . . .	 17
3.1 Boltzmann Routine . . . . .	17
3.1.1 Read Parameters Passed from CO2OSC . . . . .	17
3.1.2 Load Cross Section Data . . . . .	19
3.1.3 C-Matrix Elements . . . . .	24
3.1.4 (I-hC) Matrix . . . . .	25
3.1.5 Inversion of (I-hC) . . . . .	26

	Page
3.1.6 Iterative Calculation of Electron Number Density . . . . .	26
3.1.7 Transport Coefficients . . . . .	28
3.1.8 Pumping Terms . . . . .	28
3.2 Subprogram Interp . . . . .	29
3.3 Plotting Routines . . . . .	31
3.3.1 Input Parameters . . . . .	31
3.3.2 Linear Plot . . . . .	32
3.3.3 Axes and Tick Marks . . . . .	32
3.3.4 Label Tick Marks . . . . .	33
3.3.5 Plot the Data . . . . .	34
3.3.6 Log Plot . . . . .	35
3.4 Data Files . . . . .	35
3.5 Interface with CO2OSC . . . . .	36
3.6 User's Guide . . . . .	38
IV. Results . . . . .	41
4.1 Comparison with Published Results for a Single Gas (N2) . . . . .	41
4.2 Comparison with Published Results for a Gas Mixture . . . . .	57
4.3 The Effects of Boltz Related Parameters on Laser Output . . . . .	57
V. Conclusions . . . . .	68
VI. Recommendations . . . . .	70
Appendix A. Program Boltz . . . . .	72
Appendix B. Program CO2OSC Modifications . . . . .	86
Appendix C. Subprograms . . . . .	91

	Page
Appendix D. Data Files . . . . .	104
D.1 Nitrogen . . . . .	104
D.2 CO <sub>2</sub> . . . . .	108
D.3 Helium . . . . .	111
Appendix E. Methods of Numerical Solution . . . . .	113
E.1 Gauss-Jordan . . . . .	113
E.2 L-U Decomposition . . . . .	114
E.3 Gauss-Seidel . . . . .	115
E.4 Successive Over Relaxation (SOR) . . . . .	116
E.5 Speed and Accuracy Comparison . . . . .	117
Bibliography . . . . .	119
Vita . . . . .	121

*Abstract*

The collisional Boltzmann equation was solved numerically to obtain excitation rates for use in a CO<sub>2</sub> laser design program. The program was written in Microsoft QuickBasic for use on the IBM Personal Computer or equivalent.

Program validation involved comparisons of computed transport coefficients with experimental data and previous theoretical work. Four different numerical algorithms were evaluated in terms of accuracy and efficiency. L-U decomposition was identified as the preferred approach. The calculated transport coefficients were found to agree with empirical data within one to five percent.

The program was integrated into a CO<sub>2</sub> laser design program. Studies were then performed to evaluate the effects on predicted laser output power and energy density as parameters affecting electron kinetics were changed. Plotting routines were written for both programs.

A Numerical Solution to the Boltzmann Equation  
for Use in Calculating Pumping Rates  
in a CO<sub>2</sub> Discharge Laser

*I. The CO<sub>2</sub> Laser Model*

The CO<sub>2</sub> laser uses the vibrational-rotational energy transitions of the CO<sub>2</sub> molecule. This molecule is a linear triatomic (1:28). As such, it is subject to asymmetric stretch, symmetric stretch and bending vibrational modes (1:29), (2:25), (3:8). These vibrational motions may be denoted as CO<sub>2</sub>(100) or  $\nu_1$ , CO<sub>2</sub>(010) or  $\nu_2$ , and CO<sub>2</sub>(001) or  $\nu_3$ .

The kinetics of CO<sub>2</sub> laser has been modeled as a four level system(1:28). In this model, the  $\nu_3$  vibrational mode is the upper laser level, the  $\nu_1$  mode is the lower laser level (for 10.6 micron radiation), and the  $\nu_2$  mode is an intermediate level through which the CO<sub>2</sub> molecules must travel before reaching the ground state. In order to aid excitation of the upper laser level  $\nu_3$ , nitrogen is added. The energy difference between the  $v=1$  level for N<sub>2</sub> and the  $\nu=3$  level for CO<sub>2</sub> is 18cm<sup>-1</sup>. This energy resonance allows for rapid energy transfer from the vibrationally excited states of N<sub>2</sub> (which are nearly equally spaced (1:27)) to  $\nu=3$  of CO<sub>2</sub>. Because of its cross section, N<sub>2</sub> is readily excited in an electric discharge, and it does not radiatively relax (1:29).

In a dc electric glow discharge, electrons are accelerated by the applied field. This directed velocity is thermalized in momentum transfer collisions and causes the electrons to excite vibrational levels of CO<sub>2</sub> and N<sub>2</sub> through impact. The rate at which the various energy levels of these two species are populated is determined by the velocity distribution of the electrons. The velocity distribution of the electrons is affected by such factors as the strength of the applied field divided by the neutral gas number density (E/N), ambient gas temperature, momentum transfer and inelastic

processes as well as excited state populations. The Boltzmann equation relates all of these various factors and allows computation of the electron number density as a function of velocity (or energy). The next section develops a special case of the Boltzmann equation using assumptions appropriate to a laser. Later sections will apply it to compute pumping terms for a set of CO<sub>2</sub> laser rate equations.

## II. The Boltzmann Equation

Solving the Boltzmann equation allows us to calculate the partitioning of energy into loss channels. We will first show a general form of the Boltzmann equation and an approximation to it by way of a two term spherical harmonic expansion. We will then summarize the development of a form of the Boltzmann equation which is specific to our interests and solve it numerically.

### 2.1 Derivation of the Boltzmann Equation and Important Assumptions

We begin by noting that the number of particles in a small volume  $d\vec{x}d\vec{v}$  of phase space is

$$dn = f(\vec{x}, \vec{v}, t) d\vec{x}d\vec{v} \quad (1)$$

The function  $f(\vec{x}, \vec{v}, t)$  is the density of the particles in phase space, and is a distribution function.

The time evolution of this distribution function will be given by

$$\frac{df}{dt} = \frac{\partial f}{\partial t} + \frac{\partial f}{\partial x} \frac{dx}{dt} + \frac{\partial f}{\partial y} \frac{dy}{dt} + \frac{\partial f}{\partial z} \frac{dz}{dt} + \frac{\partial f}{\partial v_x} \frac{dv_x}{dt} + \frac{\partial f}{\partial v_y} \frac{dv_y}{dt} + \frac{\partial f}{\partial v_z} \frac{dv_z}{dt} \quad (2)$$

using

$$\frac{d\vec{v}}{dt} = \frac{\vec{F}}{m} \quad \vec{v} = \frac{d\vec{x}}{dt} \quad (3)$$

$$\frac{df}{dt} = \frac{\partial f}{\partial t} + \vec{v} \cdot \nabla_r f + \frac{\vec{F}}{m} \cdot \nabla_v f \quad (4)$$

When electrons and molecules collide, scattering will occur. This will change the electron velocity distribution function. We will write the net gain of electrons of a small volume of phase space by  $(\frac{\partial f}{\partial t})_c d\vec{x}d\vec{v}$ . The rate of change of the distribution function  $f$  as a result of scattering through electron-molecule collisions is given by  $(\frac{\partial f}{\partial t})_c$ . Therefore

$$\frac{\partial f}{\partial t} + \vec{v} \cdot \nabla_r f + \frac{\vec{F}}{m} \cdot \nabla_v f = \left( \frac{\partial f}{\partial t} \right)_c \quad (5)$$

Since we are assuming the case of a spatially uniform plasma, we have

$$\nabla_r f = 0$$

therefore

$$f(\vec{x}, \vec{v}, t) = f(\vec{v}, t); \quad (6)$$

and

$$\frac{\partial f}{\partial t} + \frac{\vec{f}}{m} \cdot \nabla_v f = \left( \frac{\partial f}{\partial t} \right)_c \quad (7)$$

An electric field exerts a force  $F$  on the electrons in a plasma such that the acceleration is equal to  $-eE/m$ . The Boltzmann equation now becomes

$$\frac{\partial f}{\partial t} - \frac{e\vec{E}}{m} \cdot \nabla_v f = \left( \frac{\partial f}{\partial t} \right)_c \quad (8)$$

Under the conditions (4:370) of a small applied electric field,  $f$  will be nearly spherically symmetric in velocity space. Since the difference between the masses of an electron and molecule is very large, an elastic collision between the two will result in a small amount of electron energy being transferred and a large deflection for the electron. We will assume that the electron mean free path is small compared to the physical dimensions of the laser tube. For a typical case of a laser tube with the neutral gas at standard temperature and pressure, and assuming an average cross section of  $5 \times 10^{-15} \text{cm}^2$ , the mean free path would be  $1/(N_m Q_m)$  which is equal to  $7 \times 10^{-6} \text{cm}$ . Most lasers would greatly exceed this dimension. The electron velocity distribution will then be nearly spherically symmetric. The distribution function may then be approximated (5:27) by the first two terms of a spherical harmonic expansion

$$f(\vec{v}) = f_0(v) + \frac{\vec{v}}{v} \cdot \vec{f}_1(v) \quad (9)$$

It will be assumed that  $|f_1| \ll f_0$ .

The collision frequency for elastic electron-molecule collisions is given by

$$\nu_c(v) = NQ_m v \quad (10)$$

where  $N$  is the neutral gas density and  $Q_m$  is the momentum transfer cross section. Using (9) and (10), equation (8) then becomes two coupled equations (3:144)

$$-\frac{\partial \vec{f}_1}{\partial t} + \frac{e\vec{E}}{m} \frac{\partial f_0}{\partial v} = NvQ_m(v)\vec{f}_1 \quad (11)$$

$$\frac{\partial f_0}{\partial t} - \frac{e}{3mv^2} \frac{\partial}{\partial v} (v^2 \vec{E} \cdot \vec{f}_1) = \left( \frac{\partial f_0}{\partial t} \right)_c \quad (12)$$

Solving this coupled pair of differential equations forms the basis for our solution to the Boltzmann equation.

## 2.2 Energy Transfer by Elastic Collisions, the Applied Electric Fields, and Inelastic Collisions

**2.2.1 Elastic Collisions** The rate at which energy will flow from the electrons to the molecules as a result of elastic collisions is given by (2:85,3:145) the energy lost by the electron ( $\Delta E$ ) multiplied by the elastic collisions cross section  $I(\theta, |\vec{v} - \vec{V}|)$ , the neutral gas density  $N$ , the electron number density  $n_0$ , the electron-molecule relative velocity  $|\vec{v} - \vec{V}|$ , the molecular velocity distribution  $f_M(\vec{V})$  and the electron velocity distribution  $f_0(v)$ , all of which is then integrated over the electron and molecular velocities ( $v$  and  $V$ ). The result will be (2:88, 3:146)

$$R = \int \int \left( \frac{m^2}{M} v^2 + m\vec{v} \cdot \vec{V} \right) Q_m(v) N n_0 f_M(V) \left[ f_0(v) + \frac{\vec{v} \cdot \vec{V}}{v} \frac{\partial f_0}{\partial v} \right] v^3 dv d\vec{V} \quad (13)$$

where  $m$  is the electron mass,  $M$  is the molecular mass, the relative velocity  $\vec{v}$  has been used for  $\vec{v} - \vec{V}$ ,  $v$  is assumed to be much greater than  $V$ .

The molecular velocity distribution function will be assumed to be Maxwellian and normalized to unity

$$\int f_M(V) d^3V = 1 \quad (14)$$

The rate of energy transfer for elastic collisions then becomes

$$R = \int \left( \frac{m^2}{M} v^5 f_o + \frac{m}{M} kT v^4 \frac{\partial f_o}{\partial v} \right) N n_o Q_m dv \quad (15)$$

where T is the ambient gas temperature.

**2.2.2 Applied Electric Field** The applied electric field is the mechanism by which the electrons in the plasma are accelerated and the excitation process is initiated. An electron with velocity  $\vec{v}$  will have a kinetic energy given by

$$\epsilon = \frac{1}{2} m v^2 \quad (16)$$

The rate of change of the kinetic energy is

$$\frac{d\epsilon}{dt} = m \vec{v} \cdot \vec{v} = -e \vec{E} \cdot \vec{v} \quad (17)$$

If we now examine the total rate of change of the electrons over all electron velocities due to the applied field, we see that

$$W_t = - \int n_o \cdot e \vec{E} \cdot \vec{v} \left[ f_o(v) + \frac{\vec{v}}{v} \cdot \vec{f}_1(v) \right] d\vec{v} \quad (18)$$

Noting that  $f_o(v)$  is an isotropic function and using the approximation (2:81)

$$\vec{v}(\vec{v} \cdot \vec{f}_1) \approx \frac{1}{3} v^2 f_1 \quad (19)$$

results in

$$W_t = -\frac{1}{3}n_o \int e\vec{E} \cdot \vec{f}_1 v^3 dv \quad (20)$$

Once steady state has been reached, the power transfer from the field to the electrons will equal power transfer from the field to the molecules (through elastic and inelastic collisions). We can then obtain power transfer through inelastic collisions only by subtracting equation (15) from (20). If we examine this result for the case of power transfer through inelastic collisions by a single electron in the velocity space element  $v^2 dv$  we obtain the relation

$$-\frac{1}{3}ev\vec{E} \cdot \vec{f}_1 - \frac{m^2}{M}f_o N Q_m v^3 - \frac{m}{M}kTN Q_m \frac{\partial f_o}{\partial v} v^2 \quad (21)$$

which may be substituted into (12) to account for the contribution due to elastic collisions. The result is

$$\frac{\partial f_o}{\partial t} - \frac{1}{mv^2} \frac{\partial}{\partial v} \left( \frac{ev^2}{3} \vec{E} \cdot \vec{f}_1 + \frac{m^2}{M} N Q_m v^4 f_o + \frac{mkT}{M} N Q_m v^3 \frac{\partial f_o}{\partial v} \right) = \left( \frac{\partial f_o}{\partial t} \right)_{inel} \quad (22)$$

Let us return for a moment to the first of our two coupled differential equations

$$-\frac{\partial \vec{f}_1}{\partial t} + \frac{e\vec{E}}{m} \frac{\partial f_o}{\partial v} = N v Q_m(v) \vec{f}_1$$

A solution to this equation (2:90, 3:144) is

$$\vec{f}_1 = \frac{e\vec{E} \frac{\partial f_o}{\partial v}}{mNvQ_m} + e^{-NvQ_m t} \quad (23)$$

Examination of the exponential term for practical laser plasma parameters shows that it can be neglected, since its relaxation time would be very fast compared to the time scale of the other processes. For example, at standard temperature and pressure we have (6:17)  $N \approx 2.69 \times 10^{19} \text{cm}^{-3}$

(Loschmidt's number). An electron at 273 K would have a thermal velocity  $v = 6.92 \times 10^8 \text{ cm/sec}$ . A typical cross section (6:39) would be  $5 \times 10^{-15} \text{ cm}^2$ . This would yield a relaxation time

$$\tau = (NvQ_m)^{-1} = 10^{-14} \text{ seconds} \quad (24)$$

We may then write

$$\tilde{f}_1 = \frac{e\bar{E} \frac{\partial f_0}{\partial v}}{mNvQ_m} \quad (25)$$

Using equation (25) to eliminate  $f_1$  from equation (22), and employing the variable substitution

$$u = \frac{mv^2}{2e} \quad (26)$$

equation (22) becomes

$$\frac{\partial f_0}{\partial t} - \left( \frac{2e}{mu} \right)^{\frac{1}{2}} \frac{\partial}{\partial u} \left( \frac{E^2 u}{3NQ_m} \frac{\partial f_0}{\partial u} + \frac{2mNQ_m u^2 f_0}{M} + \frac{2mkTNQ_m u^2}{Mc} \frac{\partial f_0}{\partial u} \right) = \left( \frac{\partial f_0}{\partial t} \right)_{inel} \quad (27)$$

**2.2.3 Inelastic Collisions** The variable  $u$  of equation (26) is expressed in units of volts.  $I$  is related to the translational energy of an electron through

$$eu = \frac{1}{2}mv^2 \quad (28)$$

We may also express the energy,  $E_j$ , of the  $j$ th internal state of a molecule by

$$E_j = eu_j \quad (29)$$

The molecular velocities are assumed to be very small in the system we are examining, compared to the electron velocities. We will therefore assume that the inelastic cross section for the excitation of the molecule from its ground state to any internal state does not depend on the

molecular velocity. The inelastic cross section for this  $j^{th}$  internal state, which we shall denote  $Q_j(u^*)$ , will thus only depend on the electron's initial energy  $u^*$  (as determined by the electron's initial velocity  $v^*$ ).

As electrons suffer inelastic collisions with and transfer energy to the molecules, the distribution function at energy  $u$  will be affected two ways. First, the density per unit volume per unit velocity space at a particular energy  $u$  will gain as electrons with an initial energy of  $u + u_j$  collide with a molecule and transition to a final energy of  $u$ . This gain is given by (3:149)

$$\left(\frac{\partial f_o}{\partial t}\right)_{gain} = \sum_j N f_o(u + u_j) Q_j(u + u_j) (u + u_j) \left(\frac{2e}{mu}\right)^{\frac{1}{2}} \quad (30)$$

Second, there is a loss at energy  $u$  as electrons with initial energy  $u$  suffer an inelastic collision with a molecule and transition to a lower final energy value. This loss is given by (3:149)

$$\left(\frac{\partial f_o}{\partial t}\right)_{loss} = \sum_j N f_o(u) Q_j(u) u \left(\frac{2e}{mu}\right)^{\frac{1}{2}} \quad (31)$$

The rate of change of  $f_o(u)$  due to inelastic processes will thus be found by subtracting equation (30) from equation (29) to yield

$$\left(\frac{\partial f_o}{\partial t}\right)_{inel} = \sum_j N [f_o(u + u_j) Q_j(u + u_j) (u + u_j) - f_o(u) Q_j(u) u] \left(\frac{2e}{mu}\right)^{\frac{1}{2}} \quad (32)$$

Electrons colliding with excited molecules may also gain energy. These collisions of the second kind in which excitation energy goes back to electron are called superelastic, and the effect on the distribution function is given by (3:149)

$$\left(\frac{\partial f_o}{\partial t}\right)_{super} = \sum_j N_j [f_o(u - u_j) Q_{-j}(u - u_j) (u - u_j) - f_o(u) Q_{-j}(u) u] \left(\frac{2e}{mu}\right)^{\frac{1}{2}} \quad (33)$$

where  $Q_{-j}$  denotes the cross section for a superelastic collision.

We may now substitute into equation (27) for the term  $\left(\frac{\partial f_k}{\partial t}\right)_{inel}$ , equations (32) and (33).

The result of this is

$$\begin{aligned} & \frac{1}{3} \left(\frac{E}{N}\right)^2 \frac{d}{du} \left(\frac{u}{Q_m} \frac{df_o}{du}\right) + \frac{2m}{M} \frac{d}{du} (u^2 Q_m f_o) + \frac{2mkT}{Me} \frac{d}{du} \left(Q_m u^2 \frac{df}{du}\right) \\ & + \sum_j [(u + u_j) f_o(u + u_j) Q_j(u + u_j) - u f(u) Q_j(u)] \\ & + \frac{1}{N} \sum_j N_j [(u - u_j) f(u - u_j) Q_{-j}(u - u_j) - u f_o(u) Q_{-j}(u)] = \left(\frac{mu}{2e}\right)^{\frac{1}{2}} \frac{\partial f_o}{\partial f} \end{aligned} \quad (34)$$

This is the Boltzmann equation for the isotropic part  $f_o(u)$  of the nearly isotropic electron energy distribution function, for a gas containing one type of molecule. When multiple species are present, it is necessary to sum over the contributions of each. The resulting equation is (3:150)

$$\begin{aligned} & \frac{1}{3} \left(\frac{E}{N}\right)^2 \frac{d}{du} \left(\frac{u}{\sum_k \delta_k Q_m^k} \frac{df_o}{du}\right) + 2m \frac{d}{du} \left(\sum_k \delta_k \frac{Q_m^k}{M_k} u^2 f_o\right) + \frac{2mkT}{e} \frac{d}{du} \left(\sum_k \delta_k \frac{Q_m^k}{M_k} u^2 \frac{df_o}{du}\right) \\ & + \sum_k \sum_j \delta_k [(u + u_{jk}) f_o(u + u_{jk}) Q_j^k(u + u_{jk}) - u f_o(u) Q_j^k(u)] \\ & + \sum_k \sum_j \delta_{jk} [(u - u_{jk}) f_o(u - u_{jk}) Q_{-j}^k(u - u_{jk}) - u f_o(u) Q_{-j}^k(u)] = \left(\frac{mu}{2e}\right)^{\frac{1}{2}} \left(\frac{\partial f_o}{\partial t}\right) \end{aligned} \quad (35)$$

where:  $k$  designates a particular molecular species,  $\delta_k = N_k/N$ ,  $\delta_{jk} = N_{jk}/N$ , and  $N_{jk}$  is the number density of the excited molecules in state  $j$  of species  $k$ . The energy of a molecule of species  $k$  in state  $j$  is given by  $u_{jk}$ .

### 2.3 Finite Difference Form of the Boltzmann Equation

A numerical solution to the Boltzmann equation based upon a finite difference approach has been developed by Rockwood (7:2348, 8:377) and expanded upon by Davies (9:369). We will now express equation (35) in finite difference form, and later use this result as the basis for our main computer program. This approach consists of two main parts. First, equation (35) is expressed in

terms of the normalized electron number density

$$n(u) = u^{\frac{1}{2}} f(u)$$

$$\int_0^{\infty} u^{\frac{1}{2}} f(u) du = 1 \quad (36)$$

Also, a variable change from  $u$ (volts) to  $\epsilon$  (electron volts) is accomplished where

$$\epsilon = eu \quad (37)$$

Second, equation (35) is converted in to a set of  $K$  - coupled ordinary differential equations. This is accomplished by dividing the electron energy axis into  $K$  cells of width  $\Delta\epsilon$ . Having finite differenced the energy axis in to  $K$  cells, we now project equation (35) on to it. The result is a set of  $K$  coupled first order differential equations, which have been approximated by a finite difference representation. The approximations made to do this are

$$\frac{dn}{d\epsilon} \approx \frac{n_{k+1} - n_k}{\Delta\epsilon}$$

$$\frac{d^2n}{d\epsilon^2} \approx \frac{n_{k-1} - 2n_k + n_{k+1}}{(\Delta\epsilon)^2} \quad (38)$$

where  $n_k$  is the electron number density in the energy range  $(k-1)\Delta\epsilon = \epsilon_k$  to  $k\Delta\epsilon$ . The Boltzmann equation thus becomes

$$a_{k-1}n_{k-1} + b_{k+1}n_{k+1} - (a_k + b_k)n_k + \sum_j \sum_s N_s (R_{js,k+m_{js}} n_{k+m_{js}} - R_{js,k} n_k)$$

$$+ \sum_j \sum_s N_s \Delta_{js} (R'_{js,k-m_{js}} n_{k-m_{js}} - R'_{js,k} n_k) = \dot{n} \quad (39)$$

where:

$$a_k = \frac{2Ne}{3m} \left( \frac{E}{N} \right)^2 \frac{1}{\nu_k^+/N} \left( \frac{1}{\Delta\epsilon} \right)^2 \left( \epsilon_k^+ + \frac{\Delta\epsilon}{4} \right) + \frac{\rho_k^+}{2\Delta\epsilon} \left( \frac{kT}{2} - \epsilon_k^+ + \frac{2kT\epsilon_k^+}{\Delta\epsilon} \right)$$

$$\begin{aligned}
b_{k+1} &= \frac{2Ne}{3m} \left( \frac{E}{N} \right)^2 \frac{1}{\nu_k^+/N} \left( \frac{1}{\Delta\epsilon} \right)^2 \left( \epsilon_k^+ - \frac{\Delta\epsilon}{4} \right) + \frac{\bar{\nu}_k^+}{2\Delta\epsilon} \left( \epsilon_k^+ - \frac{kT}{2} + \frac{2kT\epsilon_k^+}{\Delta\epsilon} \right) \\
\frac{\nu_k^+}{N} &= \left( \frac{2e\epsilon_k^+}{m} \right)^{\frac{1}{2}} \sum_s \delta_s Q_m^s(\epsilon_k^+) \\
\bar{\nu}_k^+ &= \left( \frac{2e\epsilon_k^+}{m} \right)^{\frac{1}{2}} 2mN \sum_s [\delta_s Q_m^s(\epsilon_k^+)/M_s] \\
R_{j,s,k} &= R_{j,s}(\epsilon_k^+) = Q_j^s(\epsilon_k^+) \left( \frac{2e\epsilon_k^+}{m} \right)^{\frac{1}{2}} \\
R'_{j,s,k} &= R'_{j,s}(\epsilon_k^+) = Q_{-j}^s(\epsilon_k^+) v(\epsilon_k^+) = \frac{\epsilon_k^+ + \epsilon_{j,s}^*}{\epsilon_k^+} Q_j^s(\epsilon_k^+ + \epsilon_{j,s}^*) \left( \frac{2e\epsilon_k^+}{m} \right)^{\frac{1}{2}} \\
m_{j,s} &= \text{the nearest integer to } \frac{eu_{j,s}}{\Delta\epsilon} \\
\Delta_{j,s} &= \exp(-\epsilon_{j,s}/kT_{j,s}^v) \\
\delta_s &= \frac{N_s}{N}
\end{aligned}$$

In this analysis:

$$k = 8.6174 \times 10^{-5} \text{ eV/K}$$

T = degrees Kelvin

$T_{j,s}^v$  = effective vibrational temp of excited state  $j$  of species  $s$

$\epsilon_k^+$  = value of the right hand edge of bin  $k$  in electron volts

$m, M_s$  are in kg

$Q_m^s, Q_j^s$  are in square meters

$\Delta_{j,s} = N_{j,s}/N_s$  is the fraction of a species  $s$  that is in the  $j^{\text{th}}$

excited (internal) state

$u_{j,s}$  is the energy loss in volts associated with the  $j^{\text{th}}$  inelastic process in species  $s$ .

Recalling equations (27) - (35), the grouping of terms are such that (9:98)

$a_k$  is the rate at which electrons in energy bin  $k$  transition to energy bin  $(k+1)$  due to elastic collisions

$b_k$  is the rate at which electrons in energy bin  $k$  transition to energy bin  $(k-1)$  due to elastic collisions

$R_{j,s,k+m_{j,s}}$  is the rate at which electrons in energy bin  $(k + m_{j,s})$  transition to energy bin  $k$

due to inelastic collisions

$R'_{j,s,k-m_{js}}$ , is the rate at which electrons in energy bin  $(k - m_{js})$  transition to energy bin  $k$  due to superelastic collisions

We will now apply boundary conditions to this model. The bin representing the lowest energy will have index  $k = 1$ . Since no electrons can transition to or from an energy bin below zero energy, we must have  $b_1 = 0$  and  $R'_{j,s} = 0$  for  $k - m_{js} < 1$ . The bin representing the highest energy will have index  $k=K$ . Since no electrons can transition to or from an energy bin above the maximum energy level ( $\epsilon_k^+$ ), we must have  $a_K = 0$  and  $R_{j,s} = 0$  for  $k + m_{js} > K$ . These boundary conditions were accounted for in the computer solution to equation (39).

#### 2.4 Numerical Solution of the Boltzmann Equation

Four numerical were used to solve equation (39). These were: : Gauss-Jordan, L-U Decomposition, Gauss-Seidel and Successive Over Relaxation. An explanation of each and a comparison of the of these methods is in Appendix E. L-U decomposition was found to give the best results for both speed and accuracy. The mathematical basis for these solutions is given here.

Equation (39) can be condensed to matrix form as

$$\dot{n} = \sum_l C_{kl} n_l \quad (40)$$

As shown earlier, the matrix elements of  $C_{kl}$  give the rate of transfer of electrons from one energy bin to another. It is assumed (7:2349, 8:378) that these matrix elements are constant. As can be seen from (39), the matrix  $C$  will be a banded matrix. The tridiagonal component is due to elastic scattering and the applied electric field. Elements above the principle diagonal are due to inelastic scattering, those below are due to superelastic scattering.

Equation (40) expressed as a forward finite difference approximation becomes

$$n_k(t+h) - n_k(t) = \sum_l h C_{kl} n_l(t+h) \quad (41)$$

where  $h$  is a time step. Since we are evaluating the right hand side of equation (41) at time  $(t+h)$ , we have an implicit algorithm. The advantage of an implicit algorithm over an explicit one is that the former is unconditionally stable (10:575, 11:379). Equation (41) may be rewritten such that

$$\begin{aligned} n_k(t+h) - \sum_l h C_{kl} n_l(t+h) &= n_k(t) \\ \sum_k (\tilde{I} - hC) n_l(t+h) &= n_k(t) \\ n_l(t+h) &= \sum_k (\tilde{I} - hC)^{-1} n_k(t) \end{aligned} \quad (42)$$

where  $\tilde{I}$  is the identity matrix. This gives us an iterative algorithm whereby we may compute the electron number density distribution. Accordingly, we first form the  $C$  matrix. Next, we multiply the  $C$  matrix by the time step  $h$ , and subtract this product from the identity matrix. The inverse of this is then calculated and multiplied by the last iteration of  $n_k$  in order to determine the next estimate. As such we use  $n_k^0$  to determine  $n_k^1$ , then  $n_k^1$  to determine  $n_k^2$  and so forth. This algorithm was used in both the Gauss-Jordan and L-U decomposition methods.

## 2.5 Transport Coefficients

In order to provide the required input to the laser design program CO2OSC, we must calculate the electron drift velocity and the excitation rate of certain excited states in specific species. The drift velocity is used to determine the electron number density  $N_e$  through (3:167)

$$v_{de} N_e = j \quad (43)$$

where  $j$  is the current density in the discharge tube. The product of the electron number density  $Ne$ , the inelastic excitation rate  $\chi$  and the applicable species number density ( $N_{CO_2}$  or  $N_{N_2}$ ) provides the desired pumping term for CO2OSC.

**2.5.1 Drift Velocity** Referring to equation (39) we see that the first term in each of the expressions for  $a_k$  and  $b_{k+1}$  determine the rate at which the electrons will exchange energy with the dc electric field. We will define new terms  $\bar{a}_k$  and  $\bar{b}_k$  such that

$$\begin{aligned}\bar{a}_k &= \frac{2Ne}{3m} \left(\frac{E}{N}\right)^2 \frac{1}{\nu_k^+/N} \left(\frac{1}{\Delta\epsilon}\right)^2 \left(\epsilon_k^+ + \frac{\Delta\epsilon}{4}\right) \\ \bar{b}_k &= \frac{2Ne}{3m} \left(\frac{E}{N}\right)^2 \frac{1}{\nu_k^+/N} \left(\frac{1}{\Delta\epsilon}\right)^2 \left(\epsilon_k^+ - \frac{\Delta\epsilon}{4}\right)\end{aligned}\quad (44)$$

The rate at which electrons gain energy from the field will be

$$\dot{E}_g = \sum_k (\bar{a}_k - \bar{b}_k) n_k \Delta\epsilon \quad (45)$$

This ohmic loss or Joule heating may be written as

$$\dot{E}_g = \sum_k (\bar{a}_k - \bar{b}_k) n_k \Delta\epsilon = \vec{J} \cdot \vec{E} = JE \quad (46)$$

The units for this expression are power per unit volume or joules per cubic meter per second.

Conversion to units of electron volts gives for the drift velocity

$$v_d = \frac{\sum_k (\bar{a}_k - \bar{b}_k) n_k \Delta\epsilon}{NeE} \quad (47)$$

**2.5.2 Rate of Excitation** The excitation rate for inelastic processes is given by (7:2350)

$$R_i^j = \frac{\int \sigma_{ij}(\epsilon) v(\epsilon) n(\epsilon) d\epsilon}{\int n(\epsilon) d\epsilon} \quad (48)$$

where  $\sigma_{sj}$  is the cross section for inelastic excitation of the  $j^{\text{th}}$  internal state of species  $s$ . Conversion to a formula for numerical calculation results in

$$R_s^j \approx \frac{\sum \sigma_{sj}(\epsilon_k) v(\epsilon_k) n_k \Delta \epsilon}{\sum n_k \Delta \epsilon} = \frac{\sigma_{sj}(\epsilon_k) v(\epsilon_k) n_k}{\sum n_k} \quad (49)$$

The units for this excitation rate are  $\text{m}^3/\text{s}$ .

### *III. Computer Programs*

The main goal of this project was to write a computer subprogram which would enable the program CO2OSC to generate pumping rates for use in its laser kinetic equations. The name of this subprogram is Boltz. Both CO2OSC and Boltz are written in Microsoft QuickBasic for use on an IBM personal computer or equivalent. QuickBasic automatically detects the presence of and uses a math co-processor (if installed). All run times and performance specifications reported are for an Apple IIGS computer with an Applied Engineering PC Transporter installed. This is equivalent to an 8 MHz 8086 computer with 640K of random access memory and an 8087 math co-processor installed.

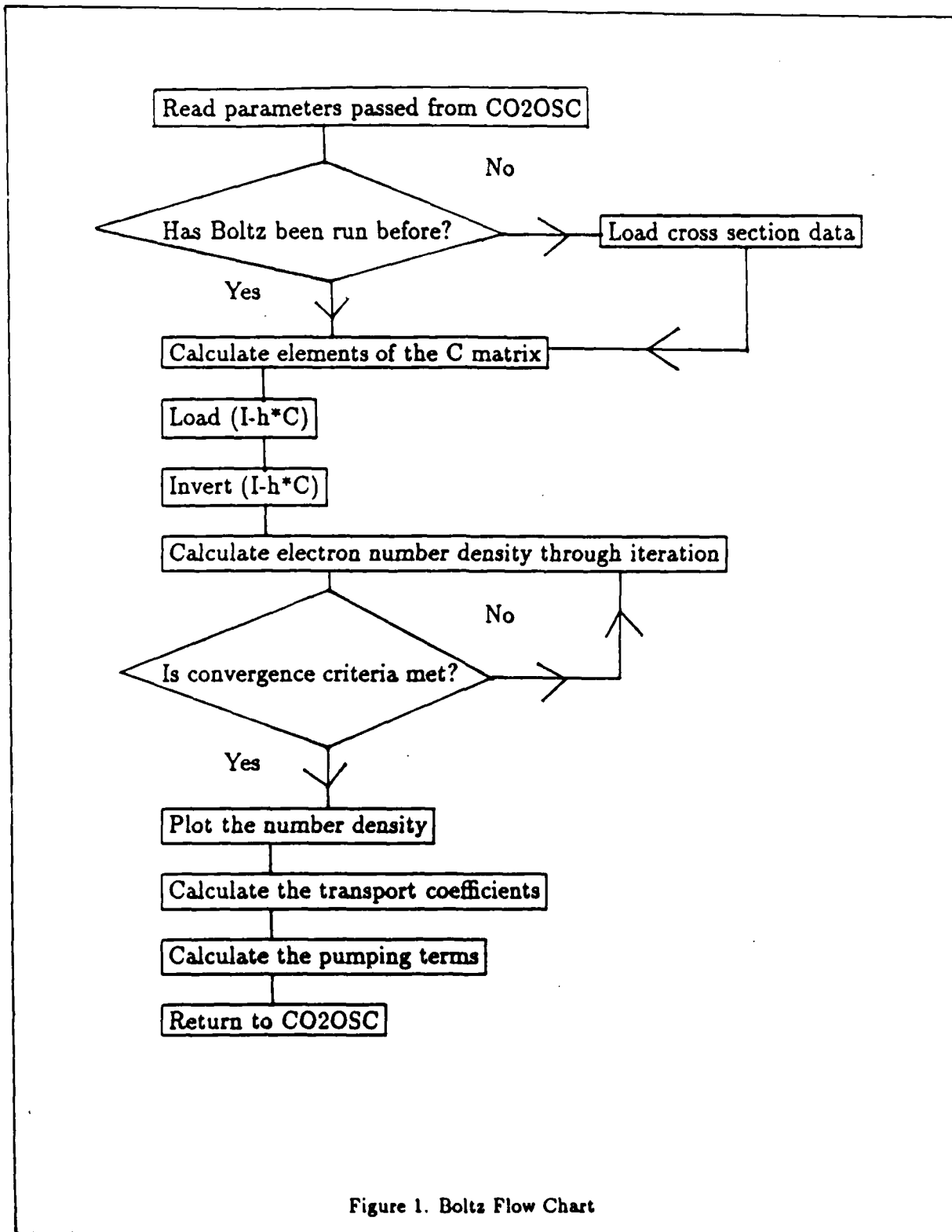
The first part of this section explains the subprogram Boltz and the second all of the subprograms it uses. The third part describes the cross section data file structure and contains instructions for user modification of the data. The fourth part lists and explains modifications to the program CO2OSC that were necessary in order to incorporate Boltz. The final part of this section is a guide to the use of CO2OSC with Boltz.

#### *3.1 Boltzmann Routine*

The purpose of the Boltz subprogram is to compute excitation rates which are then used to calculate the pumping terms for CO2OSC. Boltz does this by first calculating the electron number density distribution, and then the desired rate using equation (62). A flow chart of this subprogram is in Figure 1. Our discussion of Boltz follows this flow chart. A copy of Boltz is in Appendix A.

*3.1.1 Read Parameters Passed from CO2OSC* The first ten parameters in the argument of subprogram Boltz are all information supplied to Boltz from CO2OSC. The last three parameters in the argument of Boltz are the output Boltz supplies to CO2OSC. These parameters are:

kc% - the number of cells or bins into which the energy axis has been divided as a result



of finite differencing

de - width of a bin in electron volts

pres - total pressure of the gas mixture in atmospheres

d2 - percentage of CO<sub>2</sub> in the gas mix

d1 - percentage of N<sub>2</sub> in the gas mix

d3 - percentage of He in the gas mix

T - ambient temperature of the gas mix

VT() - a three element vector containing the vibrational temperatures for the (010), (100) and (001) states of CO<sub>2</sub>

EN - E/N, the electric field strength in volts/cm divided by the gas number density in cm<sup>-3</sup>, resulting in units for E/N of volts cm<sup>-3</sup>

jd - current density in the plasma tube in amps/cm<sup>2</sup>

Wa - pumping rate for the CO<sub>2</sub> (001) level in m<sup>-3</sup>s<sup>-1</sup>

Wb - pumping rate for the CO<sub>2</sub> (100) level in m<sup>-3</sup>s<sup>-1</sup>

Wc - weighted total pumping rate for all N<sub>2</sub> levels ( $v = 1$  to 8) in m<sup>-3</sup>s<sup>-1</sup>

*3.1.2 Load Cross Section Data* Before cross section data is read from disk, two events occur. First, Boltz checks to see if it has already been run before. If Boltz has already been previously run, then the cross section data is already in storage and need not be read again from disk. This saves about seven seconds of run time. The variable "event" is the means by which Boltz remembers if it has been previously run. By declaring the variable "event" with the command STATIC, event is not reinitialized each time the subprogram is executed. Any variable declared by using STATIC will retain its value from the previous run of the subprogram in which the variable was declared and assigned a value. Before proceeding any further, Boltz examines the value of "event". If "event" has a value not equal to one, then the subprogram Boltz has not been previously run and the program proceeds with the next step, which is to set "event" equal to one. If when evaluated the variable "event"

does have a value equal to one, then this is a signal to Boltz to skip reading in the cross section data and instead to proceed by branching to the program marker "label4:". The second action Boltz will perform is to set up storage space for the cross section data. These arrays and variables are also declared with the STATIC command so that this data, once read in from disk will not be automatically cleared from memory. While the STATIC command prevents automatic erasure of specified variables and arrays, it does not actually allocate any storage space. This is accomplished by the DIM statement.

The arrays and variables used with the cross section data are:

$eV(I,J,K)$  - energy in electron volts for a particular inelastic cross section  $K$ , of a species  $I$ , for internal energy state  $J$

$QNj(I,J,K)$  - inelastic cross section in  $cm^2$  for a particular energy  $K$ , of a species  $I$ , for an internal energy state  $J$  (where  $K = 1$  for  $N_2$  and  $K = 2$  for  $CO_2$ )

$j_{in}(I)$  - the number of internal states for a species  $K$

$K(I,J)$  - the number of pieces of cross section data in the data file for a particular species  $I$  for an internal state  $J$

$eVm1(K)$ ,  $eVm2(K)$ ,  $eVm3(K)$  - energy in electron volts for a particular momentum transfer cross section data specified by the index  $K$  for  $N_2$ ,  $CO_2$  and  $He$  respectively

$QNm1(K)$ ,  $QNm2(K)$ ,  $QNm3(K)$  - momentum transfer cross section data for  $N_2$ ,  $CO_2$  and  $He$  in  $cm^2$ , associated with the same index  $K$  as used for the  $eVm(K)$  energy arrays

$u_{js}(I,J)$  - the inelastic threshold energy in electron volts of species  $I$  for an internal state  $J$

$j_{m1}$ ,  $j_{m2}$ ,  $j_{m3}$  - the number of momentum transfer cross section data contained in the data files of  $N_2$ ,  $CO_2$  and  $He$  respectively

Other variables used during the process of reading in cross section data, but which are not static are:

$T\$$  - used to store the name of a gas as read in from the inelastic cross section data file for that gas

MT\$ - used to store the name of a gas as read in from the momentum transfer cross section data file for that gas

The actual process of reading in the cross section data file information for N2 and CO2 is essentially the same. After associating each of the data files with a device number in the OPEN statements, the first line of input is read from the data file. This first line tells Boltz how many inelastic processes are in the data file. Boltz stores this information in the jin() array, and uses it in the first loop of the data input section of the program to ensure that all of the blocks of inelastic cross section data are read from the data file for that gas. Within each block of data for each inelastic process, Boltz will read from the data file and store in ujs() the inelastic threshold energy for a particular internal energy state and store in K() the number of data points contained in the data file for a particular block of inelastic cross section data. The value for K() is then used to control the nested loop to ensure that all the data points for each block of inelastic cross section data associated with each inelastic process are read from the data file. For each inelastic cross section data point there is an energy for that point, which is stored in array eV(). The energy for each of these points is in electron volts. The cross section for each of these points is stored in the QNj() array. The cross sections, as read from the data file are in units of cm<sup>2</sup>. These are divided by 10<sup>4</sup> to convert to m<sup>2</sup> so as to be consistent with the rest of the calculations.

Momentum transfer cross section data for N2 and CO2 is read in from its respective data file immediately after the inelastic data for that gas. For He, momentum transfer cross sections are the only data read in from disk. The basic procedure for reading the momentum transfer cross section data for all three gases is essentially the same. The number of applicable data points is read in from the first line of the appropriate section of the data file and stored in jm1 for N2, jm2 for CO2, and jm3 for He. This value is then used to control the momentum transfer cross section data input loop. The energy in electron volts associated with each data point is stored in the arrays eVm1(), eVm2(), and eVm3() for N2, CO2 and He respectively. The momentum transfer cross section data

are divided by  $10^4$  so as to convert them from units of  $\text{cm}^2$  to  $\text{m}^2$ . This ends all transfer of data from the files on disk to Boltz.

The vibrational temperatures of modes (100), (010) and (001) are passed to Boltz from the program CO2OSC. These value are transferred to the VB() array for use in later calculations. All other CO2 vibrational modes are set equal to the ambient temperature. Before the elements of the C-matrix of equation (40) are calculated, required storage space is set aside and constants are defined. The storage space is declared using the REDIM statement. Arrays thus defined are dynamic. Dynamic arrays may be dimensioned using a variable. For Boltz, this user defined variable is kc%, passed to Boltz from the main program. There are two properties of the dynamic arrays used here. First, it allows the storage space to be tailored to a particular run so that unnecessary storage space isn't tied up when it might be needed elsewhere. Second, the use of dynamic arrays allows for very large arrays when compiled with the /AH option. Under normal conditions Microsoft QuickBasic is limited in storage space to 64K per array. However, the use the /AH option (required for dynamic arrays) allows a QuickBasic program to have arrays up to the limits of memory available. The ability to exceed 64K of random access memory per array was needed for some of the larger arrays reported in the results section. For an uncompiled version of CO2OSC run from the QuickBasic shell, the /AH option is available if QuickBasic is started by typing QB/AH. Versions of CO2OSC that were compiled under a QuickBasic shell that was booted with the /AH option specified, automatically have this feature built in. No special actions are required to run such a compiled version.

The arrays defined here are:

abar(I), bbar(I) - these correspond to values of a and b of equation (56) and are used to calculate the drift velocity in equation (59) and the index I refers to a particular bin

a(I), b(I) - these are the values of a and b from equation (39) where the index I refers to a particular energy bin

$R_{js}(S,J,I)$  - the rate at which electrons in energy bin  $(k+mjs)$  flow down to bin  $k$  due to inelastic collisions as detailed in equation (39), and index  $S$  represents a particular species ( $1 = N_2$ ,  $2 = CO_2$ ), the index  $J$  corresponds to a particular internal energy state, and the  $I$  index denotes the energy bin under consideration

$RT_{js}(I)$  - this array stores the sum of all the rates  $R_{js}$  over all the species and all the internal energy states for each energy bin  $I$ , as dictated by equation (39)

$RS_{js}(S,J,I)$  - the rate at which electrons in energy bin  $(k-mjs)$  flow up to energy bin  $k$  as a result of superelastic collisions as shown in equation (39), and  $S$ ,  $J$ , and  $I$  are as defined above for  $R_{js}()$

$RTS_{js}(I)$  - the sum of all the superelastic rates over all the species and all the internal energy states for each energy bin as indexed by  $I$

$C(I,I)$  - contains all the elements of the  $C$ -matrix of equation (40)

$NO(I)$  - the total electron number density in the range  $(I - 1)\Delta\epsilon$  to  $I\Delta\epsilon$  as shown in equations (40 - (42)

$NO1(I)$  - temporary storage of  $NO()$  values during the iteration part of the program

$PNO()$  - temporary single precision storage (for plotting) of the  $NO()$  values, since  $Plot1$  is single precision, and a call to that subprogram with a double precision parameter in the argument would result in a "Parameter Type Mismatch" error statement

$INDx()$ ,  $vv()$ ,  $yb()$  - all are used as temporary storage space in the  $L$ - $U$  decomposition and backsubstitution routines (10:31)

Constants defined in this section of the program are used to convert user supplied physical parameters to S.I. units and to supply values in symbolic notation for later use in calculations. The source for conversion factors is reference 6. This part of the program also converts the description of the amount of each gas contained in the mixture from percent to fraction and stores in the  $Ns()$  array the number density of each species. The order of storage in the  $Ns()$  array is  $N_2$ ,  $CO_2$ , and

He.

**3.1.3 C-Matrix Elements** This section of the program calculates the values needed to form the C-matrix of equation (40). The actual loading of this matrix is dealt with in the next section. The subroutine called *Interp* is used here. An explanation and example of its use is given in the next section.

The first item to be calculated in this section is  $mjs$  of equation (39). This represents the value of the nearest integer to  $\epsilon_{js}/\Delta\epsilon$  where  $\epsilon_{js}$  is the threshold energy loss in electron volts associated with the  $j^{th}$  internal energy state of species  $s$  and  $\Delta\epsilon$  is the bin width in electron volts. This information is stored in the array  $mjs\%(J,I)$ , where the symbol % at the end of the array name indicates that the values stored in this array are integers. The index  $J$  refers to species ( $J=1$  means  $N_2$ ,  $J=2$  means  $CO_2$ ) and  $I$  refers to a particular internal energy state (for example a particular vibrational state of  $N_2$ ). One main and one nested loop each are used for this calculation. The main loop steps through the individual species and the nested loop steps through the internal energy states of each species. The function *CINT* is internal to *QuickBasic* and provides as its output the integer nearest its argument.

The second part of this section of Boltz calculates the rates used in the C-matrix. Two main loops are used in this section. The first main loop is for the calculation of inelastic and superelastic rates. The second main loop is for the elastic rates. Within the first main loop, which uses the index  $K$  to step through the individual species of gas, are two levels of nested loops. The first level of nesting uses the index  $I$  to step through each of the individual internal energy states of each gas. This loop is controlled by the information stored in the  $jin()$  array. The second level of nesting contains two loops. The first loop uses the temporary storage arrays  $TempeV()$  and  $TepmQj()$  to store energy and inelastic cross section data for a particular internal energy state of a particular gas contained in the master three dimensional arrays  $eV()$  and  $QNj()$ . The second of these loops then uses these temporary arrays as arguments of the subprogram *Interp* in order to get a good

estimate of the inelastic cross section at a particular energy (the right hand edge of the energy bin under consideration). The second loop steps through each of the energy bins as it computes and stores the rates. The array  $Rjs()$  is used to store the inelastic rates which have been calculated for a certain species in a particular internal energy state for each energy bin. The array  $RTjs()$  is used to store the sum of the individual rates over all the internal states of all the species for a particular energy bin. This loop does likewise for the superelastic rates which it stores in the arrays  $RSjs()$  for individual and  $RTSjs()$  for total rates. The actual calculation of the inelastic and superelastic rates are as shown in equation (39). The purpose of the two IF ... THEN statements within this loop is to allow the superelastic rates for CO<sub>2</sub> (where  $K=2$ ) to be calculated with the user supplied vibrational temperatures for CO<sub>2</sub> modes (100), (010) and (001).

The last part of this section concerns calculation of the elastic rates. The calculations are as shown in equation (39). This section consists of just one FOR ... NEXT loop which steps through each energy bin with the index I. The loop is controlled by the variable  $kc\%$  (number of bins) which is supplied by the user through the main program CO2OSC. The subprogram Interp is called three times each loop in order to provide an interpolated value of the momentum transfer cross section for each of the three gases at each energy bin. The variables  $nup$  and  $nub$  represent  $\nu_k^+/N$  and  $\bar{\nu}_k^+$  respectively from equation (39).

**3.1.4 (I-hC) Matrix** This section of the program simultaneously loads the C-matrix, multiplies it by the time step  $h$ , and then subtracts this resulting product from the identity matrix. Combining these steps allows for a reduction in the number of loops which would otherwise be required. The result of this is a shorter run time for the program. The first part of this section loads into the  $a()$  and  $b()$  arrays the boundary conditions as specified in section II.3. The C-matrix is then used to store the values of  $(I-hC)$ . Most of the values can be loaded through a series of loops. Those which must be loaded outside of loops are calculated and loaded first. All calculations in this section are in accordance with equations (39) - (40).

The second part of this section consists of a single loop which loads into the (I-hC) matrix contributions along the three diagonals which are above, along and below the main diagonal. These contributions come from the a(), b(), Rjs() and PTjs() arrays. Up through this point in the program Boltz, all contributions will have been along one of these three diagonals.

The third part of this section allows for contributions to bands other than the three mentioned above. This part has one main loop and two levels of nested loops, one loop in each level. These three levels of loops step through each species of gas (N2 and then CO2, there is no inelastic contribution from He), each energy bin, and each internal state of each species. In this manner, all of the contributions to the (I-hC) matrix from the three dimensional Rjs() and RSjs() matrices are accounted for, in accordance with equations (39) and (40). Two IF... THEN statements are used in the last nested loop. These statements provide for implementation of the boundary conditions as applied to the inelastic and superelastic contributions to the (I-hC) matrix. When this section of Boltz is complete, a message is printed to the screen so as to inform the user of the progress Boltz has made thus far.

*3.1.5 Inversion of (I-hC)* Boltz now calculates the inverse of the (I-hC) matrix (which has been stored in the c() array in Boltz) in two parts. The first performs the L-U decomposition and the second part the backsubstitution described in Appendix E. Both routines were taken from reference 10, where they were published in Fortran as subroutines. They were rewritten in QuickBasic for use as a part of Boltz. They are used as part of the main code in Boltz, not as subroutines. When the inverse of the (I-hC) matrix has been completed, the inverse of (I-hC) is stored in the c() array and a message is printed to the screen informing the user of the progress of Boltz.

*3.1.6 Iterative Calculation of Electron Number Density* With the inversion of (I-hC) completed, Boltz is ready to iteratively calculate the normalized electron number density using equation (42). Before the first iteration can be calculated, an initial first guess is needed. Boltz supplies this initial guess in the form of a straight line (constant value) distribution. All elements of the NO()

array are set equal to a value of 1 before control is passed to the main iteration loop. The iteration part of Boltz consists of a main loop and two levels of nested loops. The main loop controls the maximum number of iterations that Boltz may perform. At present, this is set to 300. Of course, should the convergence criteria be met before the iteration number 300, control will pass from the iteration loop to the next part of Boltz. The first nested loop of this section also contains the second nested loop. These two loops perform the multiplication of the inverted (I-hC) matrix by the last iteration of the NO() vector. The vector result of this operation is stored in the NO1() vector. The latest iteration of the un-normalized electron number density is thus contained in NO1() at this point. It must be emphasized that what has actually been calculated by Boltz is the total electron number density within each bin, and not for a particular energy. Before the vectors NO() and NO1() can be compared to determine if the convergence criteria has been met, NO() and NO1() must both be normalized such that the sum of the elements of each is equal to 1. This will fulfill the normalization condition of equation (14). The variables "sum" and "sum1" are used to store the values needed to normalize NO() and NO1() respectively. These values are determined by summing all of the values contained in NO() (for sum) and NO1() (for sum1). Each element of these two arrays is then divided by its respective normalizing factor. The values of NO() and NO1() are then compared for each energy bin and the maximum relative difference stored in the variable cmax. This value is then compared to the convergence criteria stored in the variable conv. If cmax is less than conv, then the main iteration loop is terminated. If the convergence test fails, the NO() vector is updated with the values from NO1() and the iteration procedure is repeated.

Once the desired level of convergence is achieved, NO() is updated with the last normalized values from NO(), and plotting occurs. Since NO() is a double precision array and the arguments of Plot1 are single precision, NO() must be temporarily stored in the single precision array PNO() for plotting.

*3.1.7 Transport Coefficients* Once the normalized electron number density has been determined, Boltz calculates the transport coefficients required for output to CO2OSC. The first step in this section of the program is to calculate the values of  $\bar{a}_k$  and  $\bar{b}_k$  required in equation (56). These values are stored in the `abar()` and `bbar()` arrays.

The first transport coefficient calculated is the drift velocity. Boltz accomplishes this using equation (59) as the basis. Because the drift velocity calculation is performed in units of S.I., the value which Boltz reports is divided by .01 so as to convert the output to cm/s. Thus the reported value is in units most often found in the literature. Once the drift velocity has been computed, Boltz reports its value to the screen along with many of the original parameters that went into the calculation. This enables the user to press `SHIFT-PRINT SCREEN` and have all this information sent to the printer at once.

Boltz next calculates the inelastic excitation rates for the eight vibrational levels of N2 and modes (100) and (001) of CO2. These calculations are accomplished using equation (61). For N2, the inelastic excitation rates for each of the eight individual levels and their total are reported to the screen. For CO2, the inelastic excitation rates for (100) and (001) modes are reported. Boltz also calculates a weighted total for N2 and stores this in `TOTN2R`. This weighted sum is used in the calculation of the pumping term for N2. An approximation is made here by assuming that the eight vibrational levels of N2 are equally spaced. This means that an N2 molecule excited to the  $v=2$  state can vibrationally excite two CO2 molecules, in the  $v=3$  state an N2 molecule can excite 3 CO2 molecules, etc.

*3.1.8 Pumping Terms* The objective of Boltz is to supply CO2OSC with the pumping terms  $W_a$ ,  $W_b$  and  $W_c$ . The formula as required by CO2OSC is

$$\frac{W}{dcrit \cdot (1/tcav)} = \text{Pump Rate (sec}^{-1}\text{)} \quad (50)$$

where  $W$  can be  $W_a$ ,  $W_b$  or  $W_c$  and (3:162)

$$W_a = N_e N_{CO_2} \chi_{CO_2(001)}$$

$$W_b = N_e N_{CO_2} \chi_{CO_2(100)}$$

$$W_c = N_e N_{N_2} \chi_{N_2}$$

where  $N_e$  is the electron number density (unnormalized) and  $\chi$  is the respective inelastic excitation rate. To calculate  $N_e$ , Boltz uses (3:167)

$$N_e = (\text{current density}) / (\text{drift velocity} \times \text{electron charge})$$

This allows Boltz to complete the calculation of parameters to be passed back to CO2OSC. The final calculation performed by Boltz is energy balance. This acts as a check on the quality of the calculations performed to arrive at the electron number density.

### 3.2 Subprogram Interp

Interp is a subprogram used by Boltz to perform linear interpolation. This is important because the cross section data files contain data only at certain values of energy. An assumption is made here that linear interpolation will provide a sufficiently accurate estimate for a cross section value which falls somewhere between two known data points. This is based upon the assumption of linear behavior of the cross sections between these known points. Before Interp was written a cubic spline based interpolation program was tried. Due to the very abrupt changes in the cross section data curves (14:62-67), the cubic spline results tended to overshoot and give interpolated cross section values far removed from the known data. In some instances negative values for the cross section estimates were output by the cubic spline routine. Therefore, a linear interpolation based routine was written. A copy of the listing for subprogram Interp is in Appendix C. Interp is

based in part on the subroutine SPLINT in reference 10.

There are five parameters in the argument of Interp. The first four are input parameters supplied to Interp by the calling program, and the last is the output Interp provides to the calling program. These parameters are:

x() - an array with the known values for the abscissas

y() - an array with the known ordinates corresponding to the x() array

N - total number of points contained in the x() array

xin - the input abscissa value for which an ordinate value is to be determined by Interp

yout - the output interpolated value

The first action Interp performs is to see if the input abscissa value is within the range for which non-zero cross section values exist. If it is outside this range, this is sensed by the IF... THEN and ELSEIF statements. These statements direct Interp to set yout equal to zero and return to the main program.

If Interp has determined that a non-zero cross section value exists for xin, it then proceeds to find one. Interp uses the variables klo and khi as place markers for the low and high values in the array x(). Interp initializes these place markers with the lowest and highest index values of the x() array (1 and N). Using an IF... THEN... ELSE construction, Interp performs a looping bisection search. Interp then denotes the lower value as x(klo) and the higher value as x(khi).

Now that Interp knows the two abscissas between which xin lies, it is ready to try to obtain an ordinate to assign to xin. First though, Interp performs an internal check to make sure that it has not accidentally assigned the same point in the x() array to both x(klo) and x(khi). Once the internal error check is complete, Boltz determines the value of yout to assign to xin based upon the slope between the points (x(klo), y(klo)) and (x(khi), y(khi)). Boltz assigns to the variable h the difference in abscissa values for these two points. Even though the variable h is used elsewhere in Boltz, this does not cause a problem. Variables defined within a subprogram are local to that

subprogram. This is an advantage of defining Interp as a subprogram, instead of as a subroutine (in which variables become global). Boltz then uses h in the formula

$$y_{out} = y(klo) + ((x_{in} - x(klo))/h) * (y(khi) - y(klo))$$

The interpolated ordinate value Interp has assigned to x<sub>in</sub> is then passed back to Boltz as the parameter y<sub>out</sub>.

### 3.3 Plotting Routines

The two plotting routines used are Plot1 and Plot2. The listings for these subprograms are in Appendix C. Plot1 is used by Boltz to plot the electron number density and Plot2 is used by CO2OSC to plot laser output power and output total energy. Both subprograms provide linear and semilog plots. Plot2 has the additional capability of allowing the user to specify the minimum and maximum values for both coordinate axes and to replot a particular power or energy curve as many times as desired. Both routines make use of the subprogram cgadump (16:156), which can speed up printing out a plot on a printer (as opposed to using SHIFT-PRINT SCREEN). Since Plot1 and Plot2 are nearly identical, their descriptions are combined here, with differences highlighted as they occur.

#### 3.3.1 Input Parameters All of the parameters in the plotting subprograms are

supplied as input by the calling program. For Plot1 these are:

de - abscissa coordinate spacing

kc% - number of points to be plotted

NO() - ordinate value to be plotted

h, T, P, EN - numerical values of time step, ambient temperature, pressure, and E/N to be displayed as information on the plots

For Plot2 the parameters are:

h1, h2 - integration time steps (abscissa coordinate spacing in units of tcav)

jChStep - the point in the ordinate data file (the index number of that array point) where integration time steps are changed

ChTimeStep - if equal to zero then integration time steps are not changed (only h1 is used), and if equal to other than zero then time steps change from h1 to h2 at jChStep

tCav - unit of cavity lifetime as determined by CO2OSC and used in Plot2 to provide actual time values for the abscissa coordinates

jmax - number of points in the ordinate array to be plotted

Yquant() - ordinate array to be plotted

title\$ - character string to be displayed on the plot

**3.3.2 Linear Plot** Linear and semilog plots are generated using these routines. The code used to produce each type is very nearly the same. The linear plot code will be discussed first. The semilog discussion will only highlight differences between the two.

**3.3.3 Axes and Tick Marks** The first step of the actual plotting is to draw the horizontal and vertical axes and their corresponding tick marks. Before these lines are drawn however, the plotting routines first determine numerical ranges over which they will be working for both axes. In the case of Plot1, the minimum abscissa coordinate is the right hand edge of the first bin and the maximum is the right hand edge of the last bin. These values are stored in XMin and XMax respectively. A FOR...NEXT loop is then performed so as to sift through the NO() array and store its minimum and maximum values in the variables Min and Max. Plot2 can also automatically determine the range of values to be plotted. For the first Plot during a particular call, Plot2 will branch to the label "autocoords". The minimum and maximum values are determined in a similar manner to that of Plot1. The difference is that the minimum abscissa coordinate is zero and the

maximum corresponds to the time for the last element of Yquant(). If there is no change in time step, the maximum will be

$$XMax = (h1 \cdot tcav \cdot 10^{-9}) \cdot jmax$$

This also converts the time associated with jmax from units of tcav to nanoseconds. If a change in time step does occur, then the maximum will be

$$XMax = (jChStep \cdot h1 + (jmax - jChStep) \cdot h2) \cdot tcav \cdot 10^{-9}$$

Later in Plot2, the user may choose to replot a particular set of data and is then given the choice of manually setting these minimum and maximum values. This occurs in the section with the label "mancoords". After inputting the range from the abscissa coordinates, the user is offered the choice of letting Plot2 automatically set the range for the ordinate values or enter them manually. Manual entry affords the user the opportunity to construct plots from different computer runs which have their data displayed over the same ranges for ease of comparison.

In preparation of drawing on the screen, the routines first clear the screen with CLS 0 command. The resolution is then set to 640x200 with the SCREEN2 command. The WINDOW and VIEW commands ensure that any previous graphics commands which might affect the display are negated. The axes and tick marks are then drawn using the LINE command and two FOR...NEXT loops.

**3.3.4 Label Tick Marks** In this section of the code, tick marks have numerical values printed next to them and labels are printed in the screen. The label identifying the plot is printed first along with information concerning input parameters which may have affected the plot. The range of abscissa and ordinate values is calculated and stored in the variables XRange and YRange. The increment between abscissa and ordinate tick marks is also calculated and stored in XIncr and YIncr. These incremental values are then used in the FOR...NEXT loops which label each tick

mark.

*3.3.5 Plot the Data* Before the routines draw lines connecting data points, the pixel locations on the screen must be redefined in order to accommodate the axes. If this is not done, then the axes, tick marks and plot will bear no relationship to each other. The area where the plotting will actually occur is defined with the VIEW command, the arguments of which define the pixel coordinate location of the lower left and upper right hand corners of the plot. This lower left hand corner is set to the point where the two axes meet. The upper right hand corner is set to the corner of the screen. The pixels themselves are scaled using the WINDOW command. This allows pixel coordinates to be specified according to an arbitrary scale. This means that during plotting, QuickBasic will automatically convert the values of the coordinates of the points to be plotted to pixel coordinates and then map that point onto the screen. This eliminates the need to set up a loop to accomplish these tasks manually.

With the viewing port and pixel scaling accomplished, the routines can begin plotting. In Plot1 this is accomplished with a single FOR...NEXT loop. The abscissa and ordinate values of the points are sequentially stored in the four variables (x1,y1) and (x2,y2) and a line drawn between them. In Plot2, if no change of integration time step has occurred, then the plotting procedure is the same as that in Plot1. When a change of integration time step has been used, control in Plot2 is then transferred to the point with the label "varplot". Plotting then occurs as a two step process. Points which occur in time before the change are plotted with a spacing based upon h1. Those which occur after the change are plotted with a spacing based upon h2. For the case where one point is before the change and the other after, Plot2 applies the correct time factor to each and plots them.

When linear plotting has finished, the user is offered the opportunity to send the plot to the printer. This dump of the plot is handled by the subprogram cgadump. This routine was taken from Cooper (16:156). On an Apple IIGS with an Applied Engineering PC Transporter installed,

using SHIFT-PRINT SCREEN to dump a plot to the printer required 4 minutes 45 seconds to print the plot. Using cgdump reduced this to 2 minutes 30 seconds.

The user of Plot2 is then given the chance to replot this set of data. Choosing to do this will cause control to branch back to the label "stplot", and the minimum and maximum values of the coordinate axes may then be set manually.

*3.3.6 Log Plot* Upon completion of the linear plot, the routines perform a plot of the same data, only this time with the ordinate axis in common log scale. This part of the routines is nearly identical with the linear plot section of the code. The first change is that the ordinate values must be converted to common log values. Since QuickBasic's internal LOG command performs computation of a natural logarithm only, these log values are multiplied by .434294482. This effects conversion from natural to common logarithm. The second change occurs in determining the minimum and maximum abscissa values. For ease of interpretation of the log scale used on the ordinate axis, the minimum and maximum values are converted to integer values. This is accomplished using the INT command.

### *3.4 Data Files*

The data files are saved on disk under the names N2DATA, CO2DATA and HEDATA. These contain the cross section data for N2, CO2 and He. All are saved as sequential ASCII files. This method of storage was chosen since it would provide easy access to the data files for most users, should the data file need to be modified. Any word processor capable of reading an ASCII file may be used to change the data in these three files. It is not necessary that the user have a copy of Microsoft QuickBasic to do this. The three data files are in Appendix D.

The N2 and CO2 data files are essentially the same. The first record contains one field. This field is stored in Boltz in the array jin(). It tells how many internal energy states are contained in the data file. The second record contains three fields. The first field of this record tells how many

records there are for this particular internal energy state. The second field contains the name of the gas and the internal energy state. The third field is the threshold inelastic energy loss for this state, and is the best means for positively identifying to as which group of data in reference 14 this data set belongs. The rest of the records for that internal energy state will consist of two fields each. The first field will contain an energy in electron volts and the second its corresponding inelastic cross section in  $\text{cm}^2$ , from reference 14. After the last record for a particular internal energy state, will be the first record of the next internal state. This pattern will continue until all of the inelastic cross section data is exhausted, and the momentum transfer cross section data begins.

All three files share the same basic set up for the momentum transfer cross section data. For  $\text{N}_2$  and  $\text{CO}_2$ , this follows immediately after the inelastic record. For He, this is the only data in the file. The first record consists of two fields. The first field tells how many momentum transfer cross section records are contained in the file. The second field is the name of the gas. The rest of the records consist of two fields each. The first field is the energy in electron volts and the second its corresponding momentum transfer cross section in  $\text{cm}^2$ .

### *3.5 Interface with CO2OSC*

Boltz and its accompanying subprograms were written so as to require only minimal changes to CO2OSC. Users wishing to modify CO2OSC need to be aware however, of how Boltz fits into CO2OSC and its special requirements. Failure to allow for the requirements of Boltz could lead to a fatal error in attempting to run CO2OSC or cause erroneous output data to be generated. Listed below are all the places in CO2OSC where an impact on the main level code from Boltz can be seen. The listings for these portions of CO2OSC are in Appendix B.

One of the most important considerations for using Boltz with CO2OSC is memory requirements. Depending on the number of bins specified by the user, Boltz could temporarily need all of the remaining memory in the computer. A 640K computer can handle 150 bins when running the

compiled version of CO2OSC, and 93 bins when run from the QuickBasic shell. Because Boltz can be very memory intensive, it should be run, if at all possible, before any other subprogram. Before Boltz terminates and returns control to CO2OSC, it de-allocates all unnecessary storage.

At the beginning of CO2OSC are the declaration statements for the subprograms. Five of these subprograms pertain to Boltz or were added to CO2OSC as part of this project. Boltz and Plot2 are subprograms which are called from the main level code of CO2OSC. Interp and Plot1 are called from Boltz and cgadump is called from Plot1 and Plot2. After the last DECLARE statement is a CLEAR statement. This was added to CO2OSC so as to allocate more space to the area of memory called the stack. This was done to accommodate the extra space needed in the stack by the newly added subprograms. If more subprograms are added, the memory block set aside by the CLEAR command may have to increase. Unfortunately, stack space competes with memory requirements for string space. Setting aside too much memory for the stack may deprive some other portion of the program memory, such as string variables, space it needs to run. One possible remedy to this problem is to run CO2OSC as a compiled program, rather than from the QuickBasic shell. This will free up the memory normally occupied by the QuickBasic shell for use by CO2OSC, which is why the compiled version can be run with a higher bin setting than that available when CO2OSC is run from the QuickBasic shell.

The nominal input parameter list has been changed to include:

EN - E/N in volts cm<sup>2</sup>

VT() - vibrational temperature for CO<sub>2</sub> modes (100), (010) and (001)

kc% - number of energy bins

de - bin width in electron volts

jd - plasma tube current density in amps/cm<sup>2</sup>

Prev\$ - a string variable used to tell CO2OSC if the user has decided to use the previously computed kinetics (thereby negating the need to call Boltz again)

These same variables are again used in the main level code of CO2OSC in the sections labeled "inputroutine" and SELECT CASE Choice\$. These two sections of the code are used to display to the user the current default settings used by CO2OSC and to allow the user the opportunity to change these settings.

In the section labeled "pump rates and effective spontaneous emission rate" is where Boltz is actually called. An IF...THEN statement is used to check if the user has decided to use Boltz or the kinetic data from the previous run. After Boltz is run, a check is made to see if the user is satisfied with the Boltzmann calculation and wishes to continue the program or return to the main menu and change one or more parameters (for example bin width). If the user has chosen to continue the program, a message indicating that the laser portion of the calculation is in progress is sent to the screen. CO2OSC then uses the parameters passed to it from Boltz to calculate the pump rates to be used by CO2OSC in its rate equations.

The final change to CO2OSC is in the section labeled "Output routines". Here, Plot2 has been added. Each call to Plot2 is preceded by storage in the string variable title\$. This is a title to be displayed on the plot produced by Plot2.

### *3.6 User's Guide*

Boltz offers the user of CO2OSC the ability to account for the effects of the electric discharge excitation in computing CO2 laser output. However, it also places new demands on the user. Specifically, the user must know how to activate and use the new features of CO2OSC provided by Boltz. This section will cover the new features of CO2OSC and explain how to use them.

The procedure for booting CO2OSC is as before. However, the first screen presented to the user has eight new input parameters. These new input parameters are labeled y through y7. The results section explores the effects of these various parameters and explains the reasons why the nominal values provided in CO2OSC were chosen.

The first parameter, y, allows the user to select a value for E/N. This is the ratio of the electric field strength in volts/cm to the total neutral gas number density in  $\text{cm}^{-3}$ . Therefore the units for E/N are Volts  $\text{cm}^2$ .

The next three parameters, y1 through y3, allow the user to specify the vibrational temperatures of CO2 modes (100), (010), and (001). The unit for these vibrational temperatures is degrees kelvin.

The fifth parameter, y4, is the number of energy bins. This is the parameter that decides into how many cells or bins the energy axis will be divided. The more bins used, the better the accuracy will usually be for a particular run. There is an upward limit of 150 bins if CO2OSC is run from a compiled version, or 93 bins if run from the QuickBasic shell. The user must be aware that a larger number of bins can dramatically affect the run time for CO2OSC. As a guide, an 8MHz computer with a math coprocessor will perform a 20 bin calculation in about 30 seconds, and a 100 bin calculation in about 10 minutes.

The sixth parameter, y5, is the width in units of electron volts of an individual energy bin. Each bin has the same width. Multiplying the number of bins by the bin width gives the total energy range over which the Boltz calculations are performed.

The seventh parameter, y6, is the current density, expressed in units of amps/ $\text{cm}^2$ . This is actually a circuit parameter provided by the user. It is used by Boltz, after the electron drift velocity has been calculated, to determine the electron number density.

The eighth parameter, y7, offers the user the opportunity to use the previously computed electron kinetics, and avoid running Boltz again. This will save time if the user has decided to perform all CO2OSC runs using one particular set of variables which affect the electron kinetics, and just vary those parameters which do not affect the electron kinetics. The parameters requiring that Boltz be run again if changed are: d, e, f, g, k, y, y1, y2, y3, y4, y5, y6 and y7.

Once the final choice of parameters has been made, pressing return will start the Boltzmann

portion of the calculation. Boltz will print to the screen messages to let the user know the progress it has made. When calculation of the normalized electron number density is complete, its linear plot will be displayed. At the bottom of the screen, the user is asked if a copy should be sent to the printer. After the linear plot, a semilog version is displayed on the screen and the user is again afforded the opportunity to send a copy to the printer. After the plots have been displayed, the pertinent input parameters and calculated transport coefficients are displayed on the screen. Pressing RETURN will then present the user the chance to return to the main menu or continue the laser calculation. Choosing to continue the laser calculation causes a confirmation message to be displayed on the screen.

The only change to CO2OSC after this is the ability to now plot the laser output power and energy as a function of time from within CO2OSC. The first plot to be displayed is a linear plot of power versus time. The plotting routine automatically chooses scales for the axes for the first plot. After this plot has been displayed on the screen, the user may choose to plot this data set again. If the choice is made to replot the data, the user will then be given the choice of manually selecting or letting the program select the range over which each axis is to be plotted. After the linear plot, a semilog plot of the same data will be displayed, and the same choices offered to the user. This same cycle will then be repeated for the plot of energy versus time.

## *IV. Results*

The results presented here fall into two categories. First, a comparison is made between predictions from Boltz and the published results of others. Second, a study is made of the effects on laser output (power and energy) as various parameters related to Boltz are changed. Some of the following studies were made using the nominal parameters of CO2OSC, while others had parameters set to match the conditions for results published by others. At the end of this section are the rationale for the nominal parameters and the suggested operating limitations for CO2OSC, as these parameters and limitations apply to the Boltz modification made to CO2OSC.

### *4.1 Comparison with Published Results for a Single Gas (N<sub>2</sub>)*

In this section, comparisons of predictions from Boltz are evaluated. Data and information are also presented to gain an understanding of what happens when the number of energy bins is varied. This is an attempt to help the user of CO2OSC appreciate the tradeoffs which arise when deciding whether to use a large or small number of bins. To simplify the situation, drift velocities and excitation rates are calculated for a single gas (N<sub>2</sub>). Two sources of published results were available for comparison with Boltz. An article containing computer predictions similar in nature to Boltz was published by Rockwood and Greene (8:383,393). Their results are used here to compare with Boltz, predictions made for drift velocity and inelastic excitation rates for N<sub>2</sub>. Data for the experimental determination of electron drift velocity has been published by Crompton and Huxley (15:627-628). Comparison with their results is also made here.

The computer runs for the drift velocity data presented in Table 1 were made at a temperature of 293 K. The computer used for these and all subsequent calculations was an Apple II GS with an Applied Engineering PC Transporter, math coprocessor and 640K of ram installed. The range of E/N for the drift velocity comparisons is from five to forty Townsends (Td). Rockwood (8:383) suggested this range so as to restrict the calculations to the region where vibrational excitation

Table 1. Nitrogen Drift Velocity Comparisons

E/N ( $10^{-17}$ Vcm <sup>2</sup> )	$v_d(10^6$ cm/s)						
	Crompton	Rockwood	Error	CO2OSC(30)	Error	CO2OSC(100)	Error
5	1.095	1.09	0.5%	1.075	1.8%	1.070	2.3%
10	1.838	1.78	3.2%	1.816	1.2%	1.793	2.4%
15	2.48	2.43	2.0%	2.516	1.5%	2.46	0.8%
20	3.09	3.06	1.0%	3.18	2.9%	3.10	0.3%
30	4.17	4.21	1.0%	4.38	5.0%	4.26	2.2%
40	5.18	5.26	1.5%	5.44	5.0%	5.28	1.9%

would dominate. That restriction is followed here and is adopted as a suggested operating limitation for CO2OSC.

In Table 1, the number in parentheses following CO2OSC is the number of bins used in the calculation. The energy range was 0 to 2.5 eV for all of the CO2OSC calculations. On average, the Rockwood calculations do the best over the entire range. The 100 bin CO2OSC calculation comes in close behind. The 30 bin CO2OSC calculations would have been excellent except for the large amount of error in the higher E/N values. This table gives the user of CO2OSC an idea of some of the potential errors and magnitudes of the Boltz portion of the CO2OSC program. If the user can live with these errors, the 30 second run time of the 30 bin calculation may be more appealing than the 9.5 minute run time of the 100 bin calculation. The effect of changing the time step  $h$  on the computed drift velocity was also examined. It was found that for  $h$  equal to  $10^{-12}$  to  $10^{25}$  seconds, the drift velocity varied about .1%. Time steps above  $10^{25}$  caused the computer to overflow. Time steps below  $10^{-12}$  resulted in too many iterations, and the program aborted the calculation before the convergence criteria was met.

In addition to the above data, Rockwood also reports the results of calculations made with his program for N<sub>2</sub> at 300 K. He used a 250 point energy grid with a bin width of .01 eV. This calculation was repeated with CO2OSC using a 100 point grid with a bin width of .025 eV. The results listed in Table 2 are for electron drift velocity and inelastic excitation rates for the eight vibrational states of N<sub>2</sub>.

The drift calculations from the two different programs are in very good agreement. The

Table 2. Comparison with Rockwood 250 Point Grid			
	Rockwood	CO2OSC	Difference
$v_d(10^6 \text{ cm/s})$	1.782	1.793	0.6%
Excitation Rates			
N2 $v=1$ ( $\text{cm}^3/\text{sec}$ )	3.493E-10	3.490E-10	0.1%
2	6.532E-11	6.635E-11	1.6%
3	2.755E-11	2.824E-11	2.5%
4	4.516E-12	4.688E-12	3.8%
5	8.762E-13	9.259E-13	5.7%
6	9.911E-15	1.110E-14	12.0%
7	7.474E-16	8.616E-16	15.3%
8	0	0	0
Weighted Excitation	5.851E-10	5.899E-10	0.82%

weighted excitation is calculated by multiplying each excitation rate by its vibrational state number, and then summing the resulting products. As explained in section V, the excitation of the upper laser level of CO2 occurs in part as a result of the transfer of vibrational energy from N2. It is assumed here that the eight vibrational states on N2 are equally spaced, and that the number of CO2 molecules excited by a particular N2 molecule is directly proportional to the vibrational state of the N2 molecule. Hence, the comparison of the weighted excitations. It is interesting note that, even though the difference between the rates calculated by the two programs grows as the vibrational state increases in energy, the actual value of the inelastic excitation rates shrinks very quickly with increasing energy. This causes the weighted excitation rates to actually be much closer than might otherwise appear. Because the excitation rates calculated by Boltz play a major role in the calculation of the pump rates to be supplied to CO2OSC, it is encouraging that they are in good agreement with the published results of a commercially available program.

The results of this next section examine how the range of the energy axis used affects the results from Boltz as the number of bins is held constant at 100. The information presented here is intended to help the user of CO2OSC decide on a suitable bin width, since that is one of the user definable parameters of CO2OSC. This number of bins was chosen since from Table 1 it appeared to give better overall results. As in the previous part of this section, a single gas (N2) was used. In the next section all work will concentrate on using a typical laser mixture. The calculations reported

Table 3. Five Townsends Comparison

E/N (Td)	Bin Width(eV)	Max Energy(eV)	CO2OSC $v_d$	Energy Balance	Crompton $v_d$	Difference
5	.015	1.5	1.136	2.68E-05	1.095	3.74%
	.02	2.0	1.081	3.38E-04		1.27%
	.025	2.5	1.083	4.53E-04		1.10%
	.03	3.0	1.085	5.37E-04		0.91%
	.035	3.5	1.074	1.24E-06		1.92%
	.04	4.0	1.076	3.91E-08		1.74%
	.045	4.5	1.070	4.56E-07		2.28%
	.05	5.0	1.093	8.93E-04		0.18%
	.055	5.5	1.079	8.31E-06		1.46%
	.06	6.0	1.098	1.06E-03		0.27%
	.065	6.5	1.072	1.32E-07		2.10%
	.07	7.0	1.089	3.73E-07		0.55%
	.075	7.5	1.103	1.33E-03		0.73%
	.08	8.0	1.118	2.40E-03		2.10%

Table 4. Ten Townsends Comparison

E/N (Td)	Bin Width(eV)	Max Energy(eV)	CO2OSC $v_d$	Energy Balance	Crompton $v_d$	Difference
10	.015	1.5	2.096	2.97E-06	1.838	14.04%
	.025	2.5	1.783	1.35E-04		2.99%
	.035	3.5	1.785	3.06E-06		2.88%
	.045	4.5	1.786	1.32E-06		2.83%
	.055	5.5	1.805	1.13E-06		1.80%
	.065	6.5	1.802	1.18E-07		1.96%
	.075	7.5	1.840	3.77E-04		0.11%
	.085	8.5	1.811	6.06E-07		1.47%

here are from the subprogram Boltz, using CO2OSC as a shell to run Boltz. This was accomplished by setting the percent of N2 at 100, all other gases at 0, and using the nominal settings of parameters "a" through "x". The value for E/N was set to five, ten and forty Townsends ( $10^{-17} \text{Vcm}^2$ ). This was chosen since good agreement was achieved over this range of E/N in the previous section with other published results. The parameter for bin width was varied so that a desired energy range was achieved. The remaining parameters were left set at the nominal values. Table 3 lists the drift velocities and energy balances obtained.

The table with the smallest deviation from the experimental drift velocity over all energy ranges is Table 3, which is also the smallest E/N. It maintains an error of about 1.5%. The trend is for a worsening of the deviation within all of the tables at the extreme values of energy range. The

Table 5. Twenty Townsends Comparison						
E/N (Td)	Bin Width(eV)	Max Energy(eV)	CO2OSC $v_d$	Energy Balance	Crompton $v_d$	Difference
20	.015	1.5	4.086	1.71E-05	3.09	32.23%
	.025	2.5	3.097	3.55E-05		0.23%
	.035	3.5	3.090	2.85E-05		0 %
	.045	4.5	3.107	2.11E-06		0.55%
	.055	5.5	3.135	4.69E-07		1.46%
	.065	6.5	3.148	1.67E-07		1.88%
	.075	7.5	3.197	1.06E-07		3.46%

Table 6. Forty Townsends Comparison						
E/N (Td)	Bin Width(eV)	Max Energy(eV)	CO2OSC $v_d$	Energy Balance	Crompton $v_d$	Difference
40	.015	1.5	8.116	3.52E-07	5.18	56.68%
	.02	2.0	4.640	3.61E-06		10.42%
	.025	2.5	5.285	1.04E-05		2.03%
	.035	3.5	5.295	6.82E-09		2.22%
	.045	4.5	5.342	1.00E-06		3.13%
	.055	5.5	5.371	2.25E-06		3.69%
	.065	6.5	5.403	1.76E-07		4.31%
	.075	7.5	5.458	3.35E-05		5.37%

energy balance is very good in all of the runs. Rockwood (8:380) reports an energy balance of  $10^{-4}$  or better. Only one run failed to meet or exceed this figure. Unfortunately, energy balance does not appear to be a good predictor of drift velocity computation accuracy. Good estimates and bad estimates of drift velocity all have relatively good energy balances. Some method of establishing the credibility of a particular drift velocity estimate is needed. The motivation for finding a reliable indicator is that it would help the user of CO2OSC decide if a particular combination of energy bin width and number of bins was a good choice. This is important since drift velocity is used to calculate the total electron number density, and this in turn is used to calculate the pumping rates. In the cases presented above the true answer is known in advance and such an indicator is not needed. However, in those cases where experimental data may not be available due to a different E/N value, different temperature, or mixture of gas being used, such an indicator would be very useful.

A better indicator appears to be the normalized electron number density distribution. Figures

Electron No. Density Number of bins = 100  $\Delta\epsilon = 0.0150\text{eV}$   $h = 10.000\text{E}-08$   
 Linear Plot Temp = 300 Pressure = 1 atm  $E/N = 1.000\text{D}-16 \text{ V cm}^2$

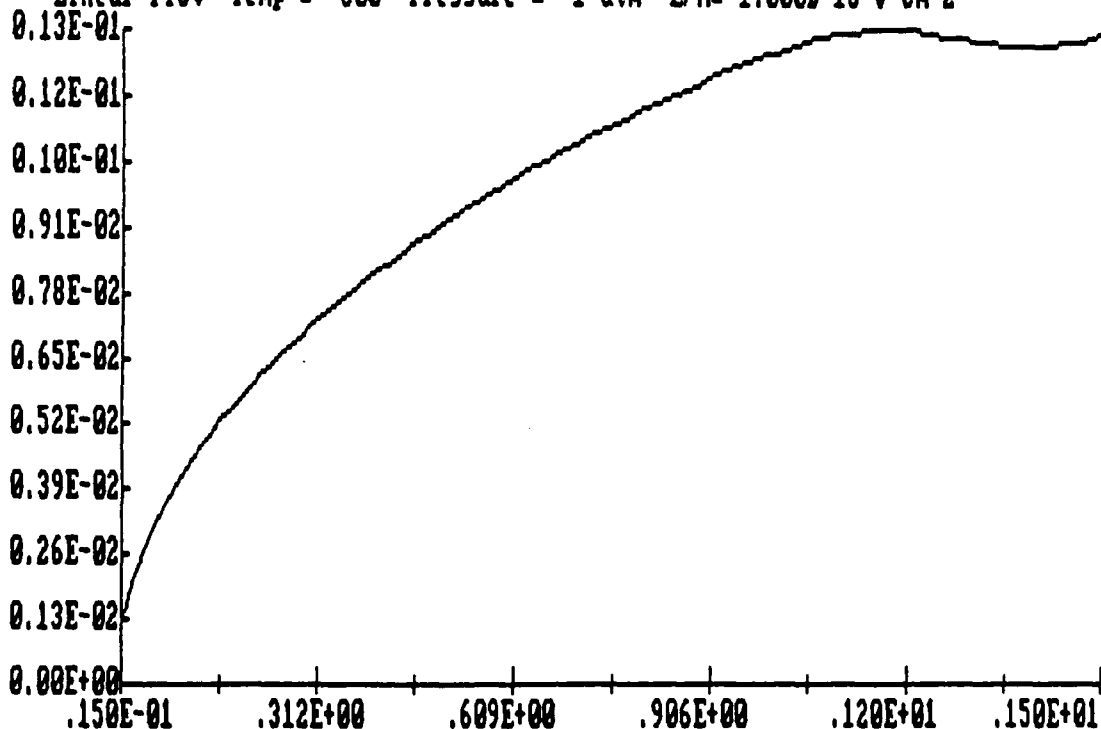


Figure 2. Ten Td Linear Plot for  $\Delta\epsilon = .015 \text{ eV}$

2 and 3 show the linear and semilog plots of the distribution for  $\text{N}_2$  with  $E/N$  equal to 10 Td and a bin width of .015 eV. As can be seen, the linear plot of the distribution does not approach zero at the higher energy bins. It appears to pass through a minimum and start to rise again. Based upon other experiences with Boltz, this is usually a good sign of impending divergence. In this case the estimate of the electron number density is unreliable, as are any calculations based upon it.

For the case where inelastic collisions are negligible and the elastic collisions occur with a constant collision frequency, a pure Maxwellian would result (17:140). While it is expected that the inelastic processes present will drive the distribution curve away from a pure Maxwellian form, the distribution should still go to zero at higher energies. It is apparent since the curve for the distribution in Figure 2 does not follow the form of a decreasing function at higher energies, an energy range of 1.5 eV is inappropriate. The same basic behavior can be seen in Figure 3.

Electron No. Density Number of bins = 100  $\Delta\epsilon = 0.0150\text{eV}$   $h = 10.000\text{E}-08$   
Semi-Log Plot Temp = 300 Pressure = 1 atm  $E/N = 1.0000\text{E}-16$  V  $\text{cm}^2$

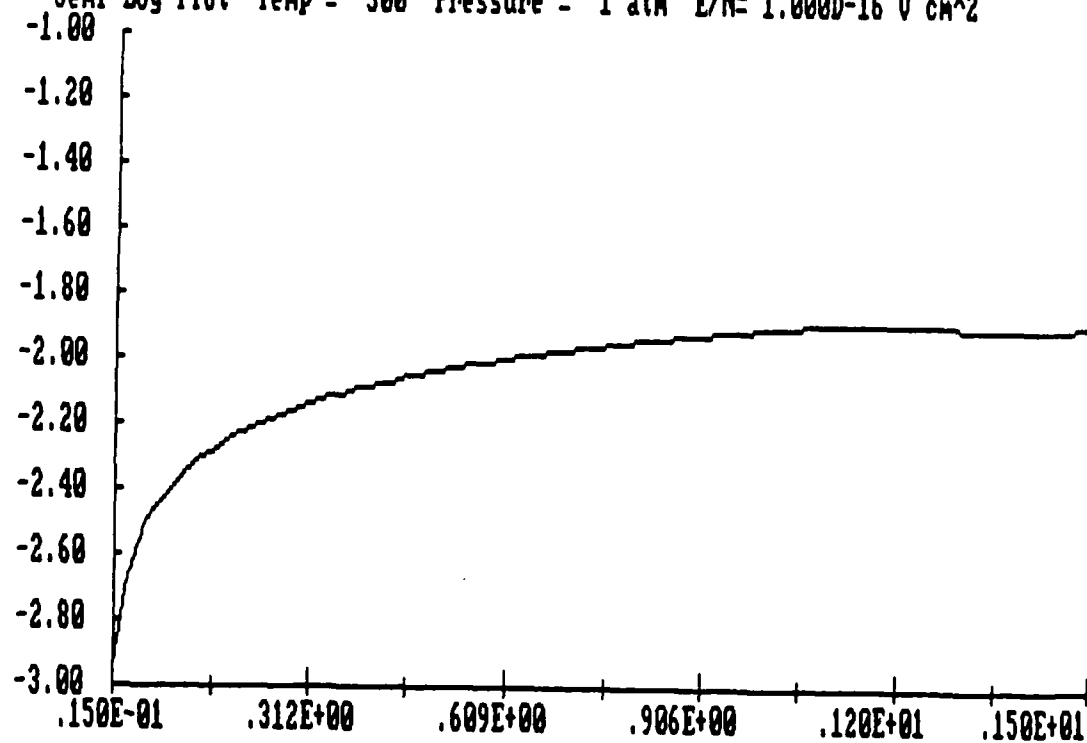
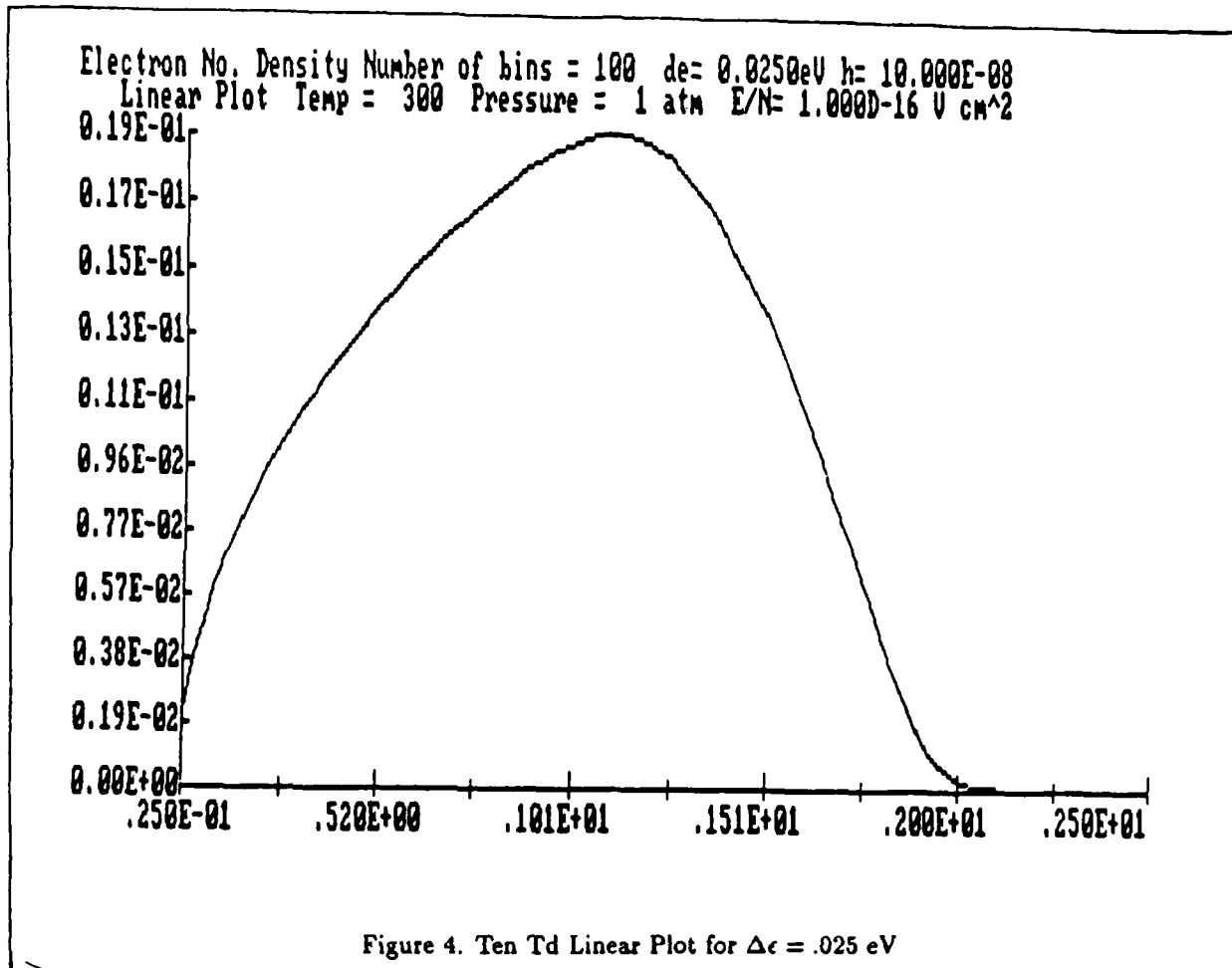


Figure 3. Ten Td Semilog Plot for  $\Delta\epsilon = .015$  eV



In Figures 4 and 5 are shown the linear and semilog plots of the distribution for  $N_2$  with  $E/N$  equal to 10 Td and a bin width of .025 eV. The plot in Figure 4 shows a distribution with the proper form. Comparison of the first two entries of Table 4 show a marked improvement in the computational value of the drift velocity when the bin width of .025 eV is used. Examination of Figure 5 shows that there is still possibly some distribution information present at higher energies.

Figures 6 and 7 show the linear and semilog plots for a bin width of .035 eV at 10 Td  $E/N$ . As can be seen from Figure 6, it is impossible to tell from the linear plot what the distribution function values are at energies above about 2 eV. The semilog plot of Figure 7 however reveals that the rate at which the distribution decreases has begun to level off. The range of values for the plot in Figure 7 covers about ten decades. Figures 8 and 9 show a continuation of this trend as the bin width is increased to .045 eV. The range of distribution presented in Figure 9 is about eleven

Electron No. Density Number of bins = 100  $\Delta\epsilon = 0.0250\text{eV}$   $h = 10.000\text{E}-08$   
Semi-Log Plot Temp = 300 Pressure = 1 atm  $E/N = 1.0000\text{E}-16$  V  $\text{cm}^2$

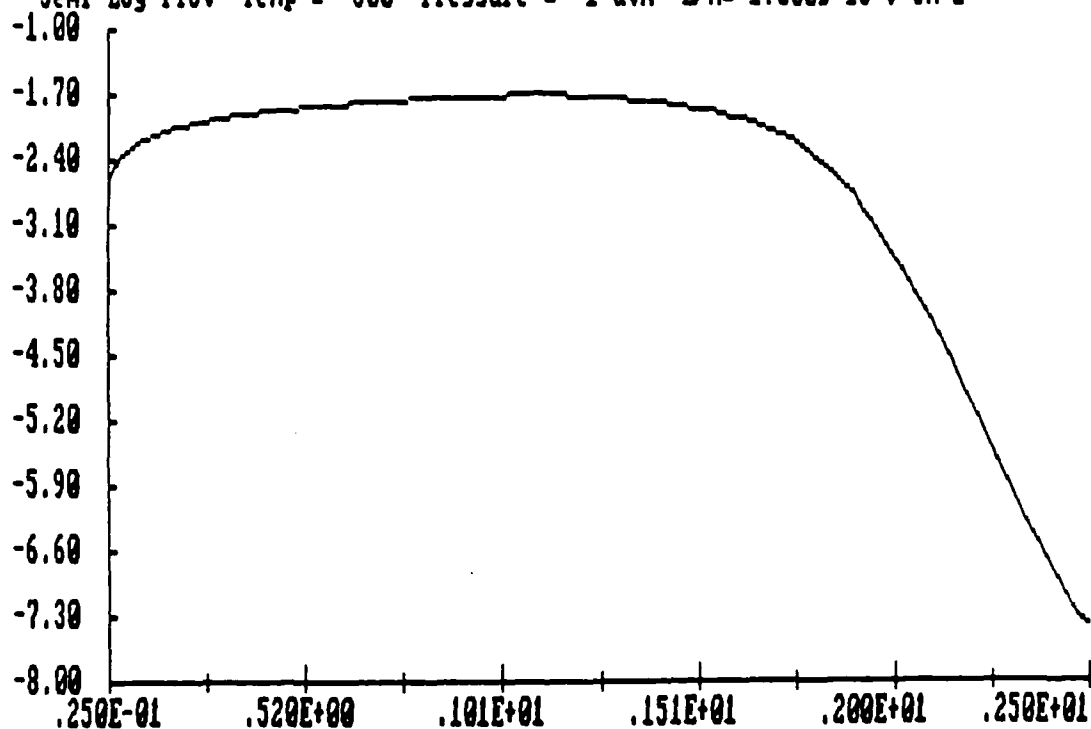
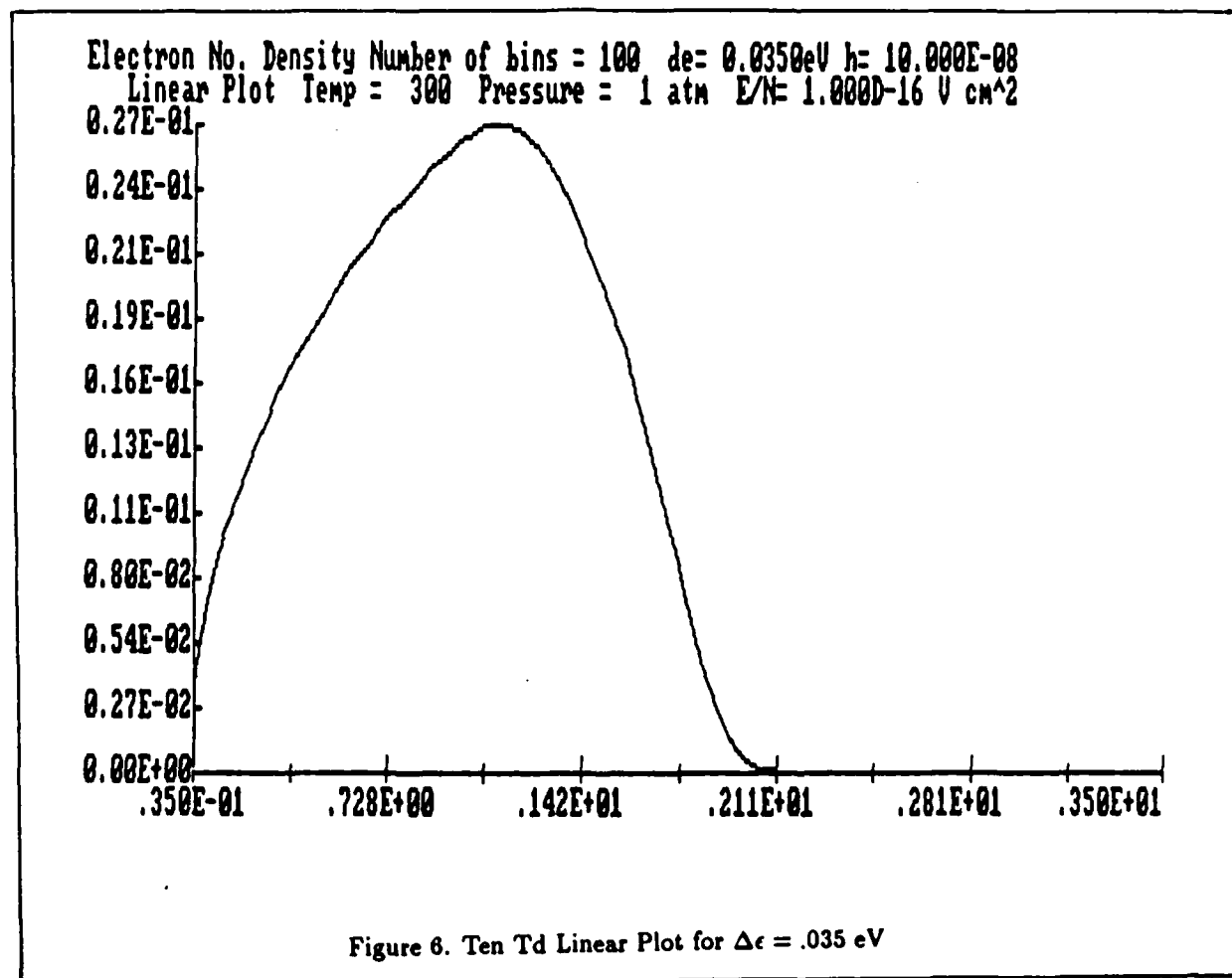


Figure 5. Ten Td Linear Plot for  $\Delta\epsilon = .025$  eV



Electron No. Density Number of bins = 100  $\Delta\epsilon = 0.0350\text{eV}$   $h = 10.000\text{E}-08$   
 Semi-Log Plot Temp = 300 Pressure = 1 atm  $E/N = 1.0000\text{-}16 \text{ V cm}^2$

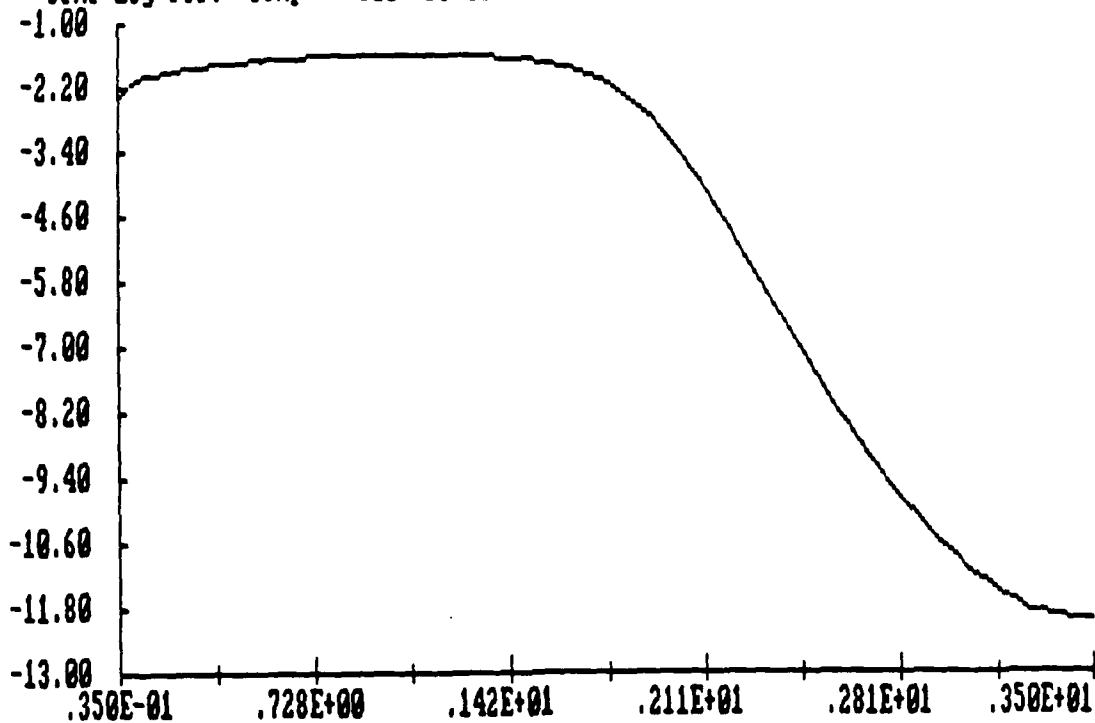


Figure 7. Ten Td Linear Plot for  $\Delta\epsilon = .035 \text{ eV}$

Electron No. Density Number of bins = 100  $\Delta\epsilon = 0.0450\text{eV}$   $h = 10.000\text{E}-08$   
Linear Plot Temp = 300 Pressure = 1 atm  $E/N = 1.0000\text{E}-16$  V  $\text{cm}^2$

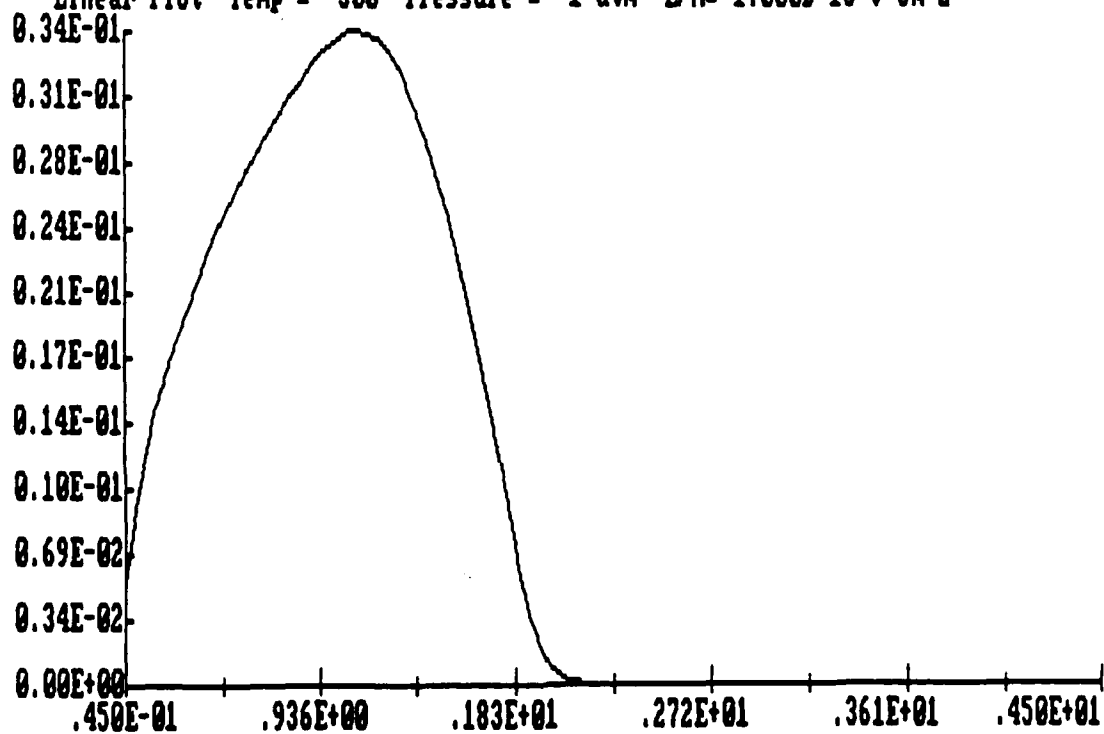


Figure 8. Ten Td Linear Plot for  $\Delta\epsilon = .045$  eV

Electron No. Density Number of bins = 100  $\Delta\epsilon = 0.0450\text{eV}$   $h = 10.000\text{E}-08$   
Semi-Log Plot Temp = 300 Pressure = 1 atm  $E/N = 1.0000\text{E}-16 \text{ V cm}^2$

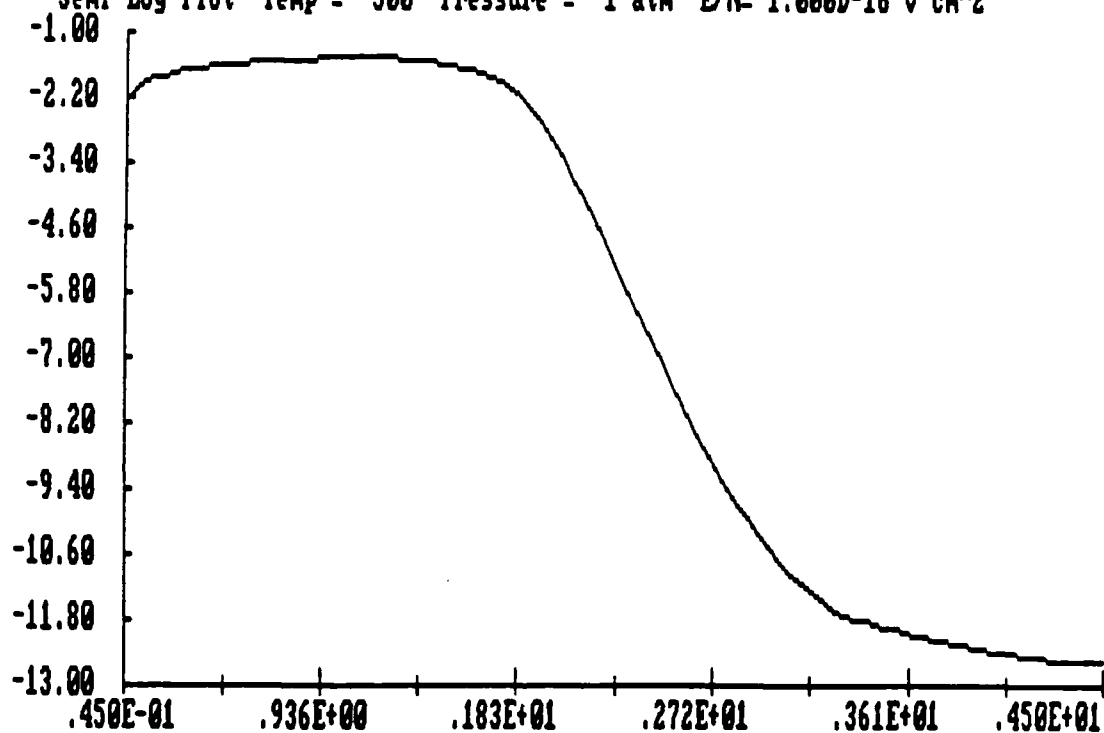


Figure 9. Ten Td Linear Plot for  $\Delta\epsilon = .045 \text{ eV}$

Electron No. Density Number of bins = 100  $\Delta\epsilon = 0.0850\text{eV}$   $h = 10.000\text{E}-08$   
 Linear Plot Temp = 300 Pressure = 1 atm  $E/N = 1.0000-16 \text{ V cm}^2$

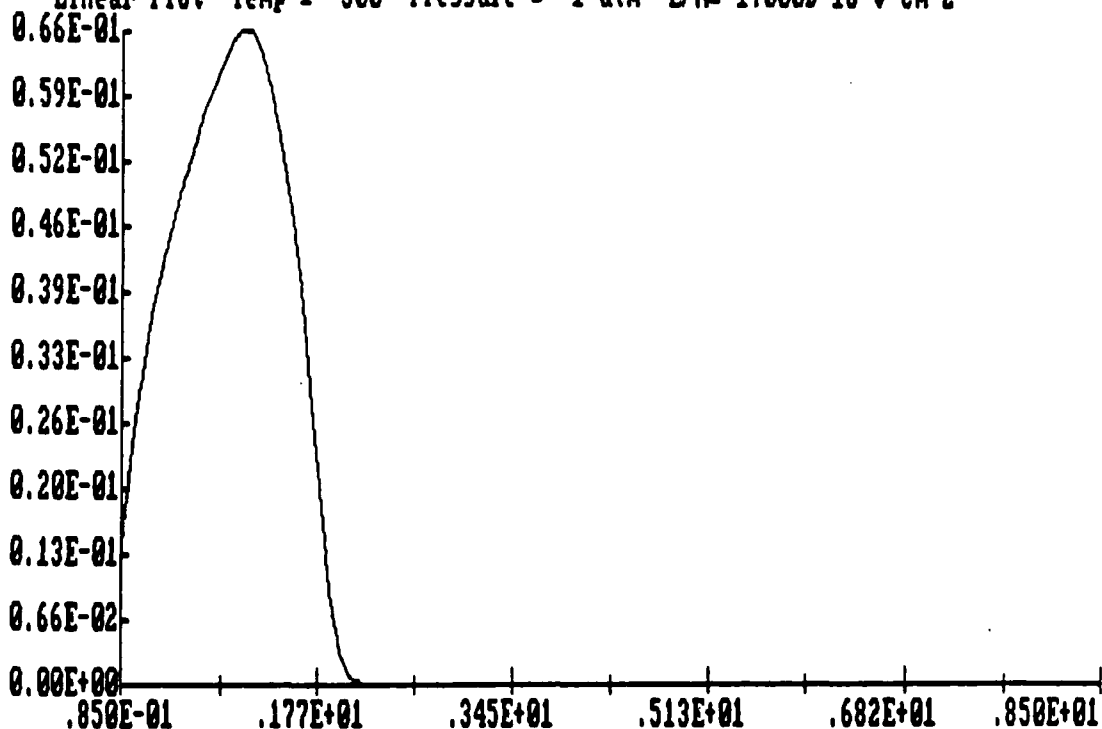


Figure 10. Ten Td Linear Plot for  $\Delta\epsilon = .085 \text{ eV}$

decades, and the drift velocity error in Table 4 also shows a slight decrease.

The error for the final bin width of Table 4 shows an increase in error. Examination of the linear plot in Figure 10 reveals that the distribution is squeezed far to the left by this particular sampling interval. The semilog plot in Figure 11 shows that in the region above about 5.5 eV the distribution begins to exhibit a slight oscillatory behavior. Previous testing with Boltz showed that if bin width was too large, a very large oscillatory behavior would be present in the calculated number density distribution. The presence of this behavior in Figure 11 is probably a good indication that the bin width is too large and should not be used. Table 7 presents the drift velocity error as a function of the number of decades from the distribution maximum to its minimum, for a particular bin width.

While the data of Table 7 presents some interesting trends for a particular  $E/N$ , there is

Electron No. Density Number of bins = 100  $\Delta\epsilon = 0.0850\text{eV}$   $h = 10.000\text{E}-08$   
Semi-Log Plot Temp = 300 Pressure = 1 atm  $E/N = 1.000\text{D}-16 \text{ V cm}^2$

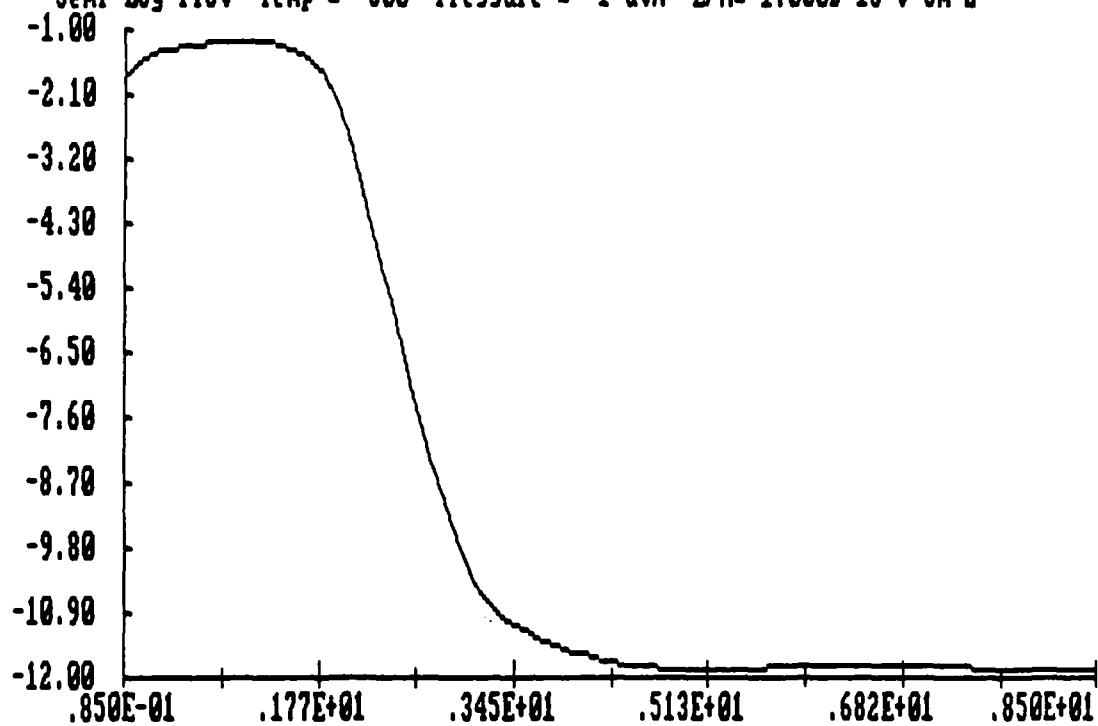


Figure 11. Ten Td Linear Plot for  $\Delta\epsilon = .085 \text{ eV}$

Table 7. Drift Velocity Error Versus Data Range

E/N (Td)	Bin Width	$v_d$ %Error	Decades of Data	E/N (Td)	Bin $v_d$ Width	Decades of %Error	Data
5	.015	3.74	0.9	10	.015	14.04	1.0
	.02	1.27	3.1		.025	2.99	5.6
	.025	1.10	10.4		.035	2.88	10.2
	.03	0.91	17.6		.045	2.83	10.9
	.035	1.92	20.7		.055	1.80	11.2
	.04	1.74	21.9		.065	1.96	11.3
	.045	2.28	22.6		.075	0.11	11.4
	.05	0.18	26.2		.085	1.47	10.8
	.055	1.46	22.6				
	.06	0.27	21.9				
	.065	2.10	21.7				
	.07	0.55	21.6				
	.075	0.73	21.4				
	.08	2.10	20.9				
E/N (Td)	Bin Width	$v_d$ %Error	Decades of Data	E/N (Td)	Bin $v_d$ Width	Decades of %Error	Data
20	.015	32.23	0.9	40	.015	56.68	1.1
	.025	0.23	2.6		.02	10.42	1.1
	.035	0	4.9		.025	2.03	1.2
	.045	0.55	5.1		.035	2.22	2.3
	.055	1.46	5.2		.045	3.13	2.3
	.065	1.86	5.2		.055	3.69	2.3
	.075	3.46	5.3		.065	4.31	2.4
					.075	5.37	2.3

no one value with regards to the range from maximum to minimum that will work in all cases. However, some general guidelines do appear to be applicable. First, too small a value of bin width, resulting in too small of an energy range should be avoided, or gross errors in the calculated number density distribution will result. The energy range should not be less than 2.5 eV. Likewise, too large of a bin width can also result in an erroneous calculated number density. A safe upper limit of 5.5 eV appears to be effective. For E/N, a range of from 5 to 40 Td can be used. These limits also make sense since electronic excitation and ionization are not modeled here. While all of these limits appear to provide a good starting point, the user of CO2OSC should always examine the linear and semilog plots so as to detect any signs of error in the calculations.

Table 8. Comparison of Predictions for Drift Velocity and Excitation  
Elliott Predictions                      CO2OSC Predictions

E/N (Td)	$v_d$	Excitation	$v_d$	Diff	Excitation	Diff
	( $\times 10^6$ cm/s)	( $\times 10^{-9}$ cm <sup>3</sup> /s)	( $\times 10^6$ cm/s)		( $\times 10^{-9}$ cm <sup>3</sup> /s)	
10	3.65	4.35	3.58	1.92%	4.39	0.92%
20	5.30	5.25	5.23	1.32%	5.31	1.14%
30	6.70	5.50	6.90	2.99%	5.56	1.09%
40	8.20	5.60	8.50	3.66%	5.71	1.96%

#### 4.2 Comparison with Published Results for a Gas Mixture

In the previous section, the transport coefficients calculated by the Boltz subprogram of CO2OSC gave good agreement with the published results available for a single gas. The case of a typical laser mixture is considered here. For this purpose, we use the nominal parameters of CO2OSC. These settings are ten percent N<sub>2</sub>, ten percent CO<sub>2</sub> and eighty percent He. The predictions made by CO2OSC are compared with the computer predictions published by Elliott (18:21,31), which were made with a program based in part on the work of Rockwood. Comparisons are made here between the computed electron drift velocities and excitation rates for the (001) mode (upper laser level) of CO<sub>2</sub>. The results are listed in Table 8.

We see in Table 8 excellent agreement between the predictions made by the two programs. We may now have confidence that CO2OSC can give reasonably good predictions of the transport coefficients we need for a gas mixture, as well as for a single gas. We can therefore be certain within the error bounds listed above for excitation, that Boltz does in fact provide the correct information upon which the pump rate equations are based.

#### 4.3 The Effects of Boltz Related Parameters on Laser Output

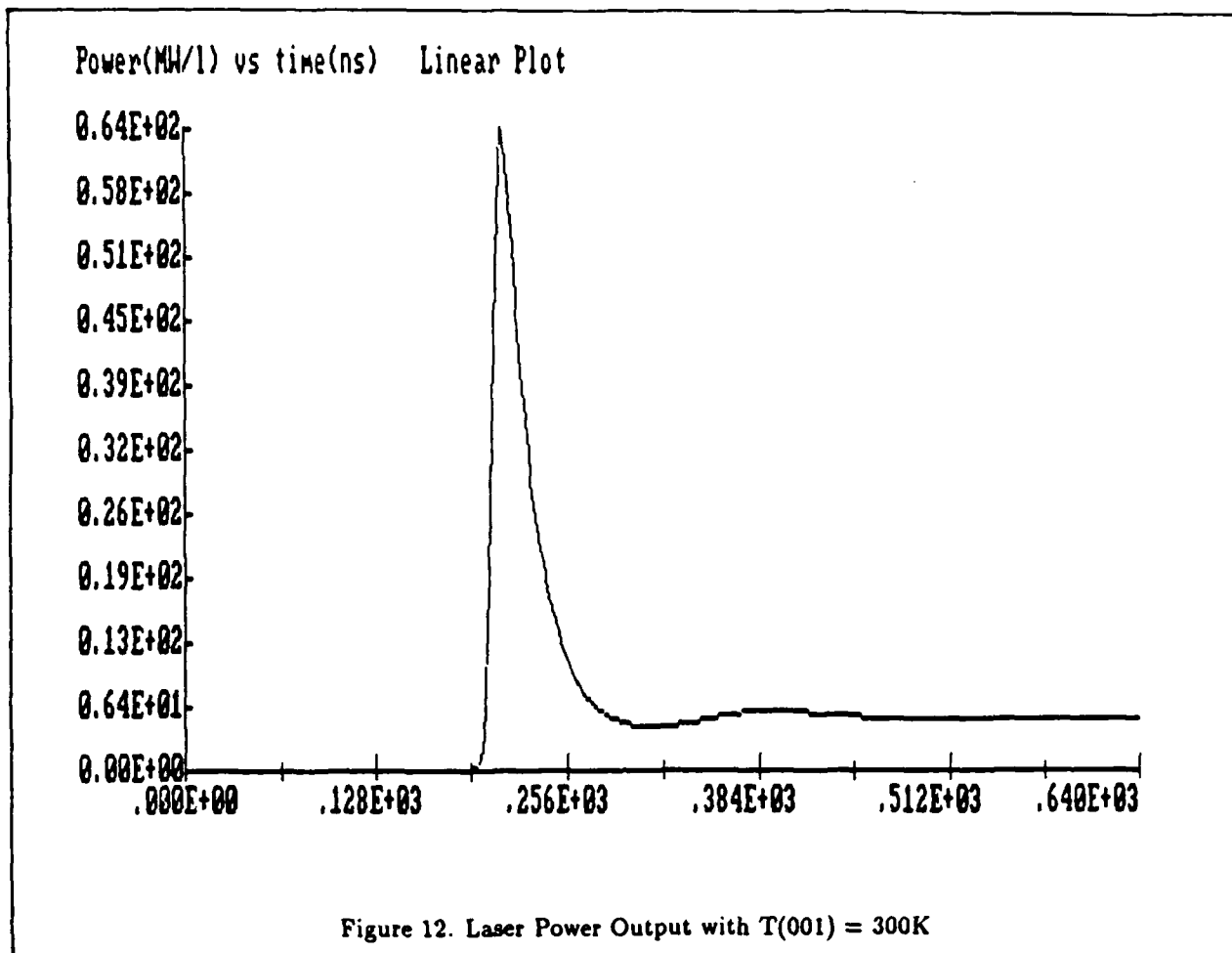
We have examined in the last two parts of this section the reliability of Boltz in providing CO2OSC the information needed to calculate pumping rates based upon a numerical solution of the Boltzmann equation. This section examines the effects on laser output power and energy of the parameters in CO2OSC that are directly related to Boltz. In particular, the parameters  $y$  through

Table 9. Comparisons of Laser Output				
T(001) (K)	Pulse Length (nsec)	Number of Bins	Peak Power (MW/l)	Total Energy density (J/l)
300	200	20	71	15
		30	71	14
		50	64	13
		100	64	13
	400	20	73	32
		30	72	29
		50	68	27
		100	69	27
4000	200	20	80	19
		30	81	19
		50	77	17
		100	76	17
	400	20	80	40
		30	81	39
		50	77	35
		100	81	39

y7 were added because of Boltz. These parameters are presented to the user in the opening screen of CO2OSC, where they may be changed to whatever values are needed. We will now change the values of some of these parameters from the nominal supplied values and see the effects as measured by the laser output power and energy calculations of CO2OSC.

The first issue addressed here is that of laser output sensitivity to bin number. This is intended to help the user of CO2OSC to see how the number of bins influences the predicted performance of a particular CO2 laser system. Computer runs at bin numbers 20, 30, 50 and 100 were made at vibrational temperatures of 300K and 4000K for CO2 mode (001). This was done for pulse lengths of 200 and 400 nanoseconds. A summary of the resulting laser output power and energy for each case is presented in Table 9.

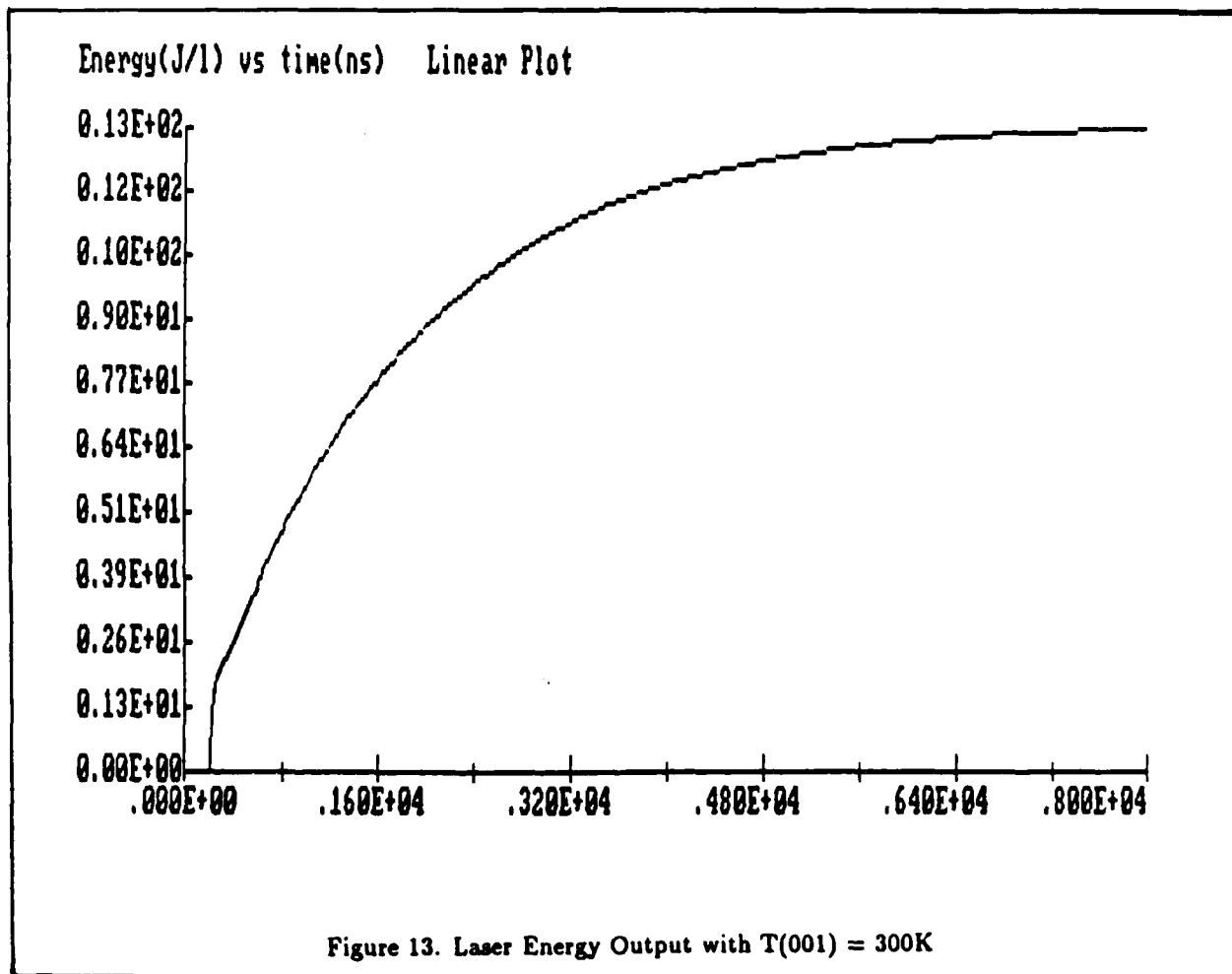
The effect of varying the bin number (for a constant energy range) is fairly consistent throughout all of these runs. There is for the most part a decrease in both the power and energy as computed by CO2OSC as the number of bins increases. The range of variations is from 5% to 11% in power and from 12% to 15% in energy. The exception is the 400 nsec run for T(001) = 4000 K. Power and energy remain fairly constant at all bin numbers. Assuming that as was previously shown the 100

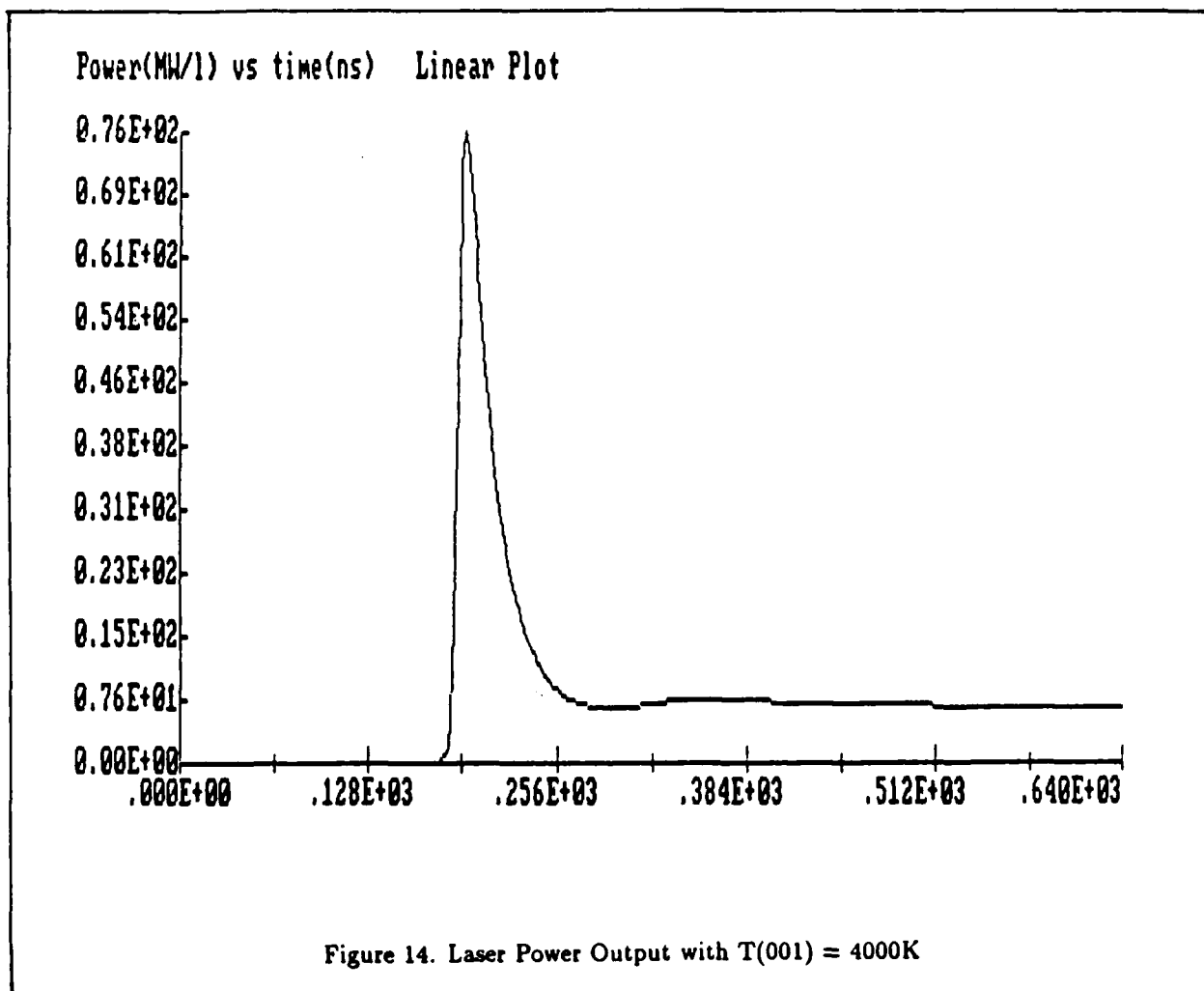


bin run is most accurate, an approximation of the error introduced by using a smaller number of bins is available from Table 9. If the answer sought by the user is one which must predict power and energy levels with great accuracy, then computer runs with a higher number of bins are in order. If however the data from the computer runs will only be used to establish a trend as a particular parameter is varied over some range, a lower bin number may be acceptable.

Examination of Table 9 shows both peak power and total energy density are higher for the system with the higher effective vibrational temperature for mode (001). This may be due to an increase in pump rates, caused by superelastic collisions. A comparison of output power and energy for the 300 K and 4000 K levels at a pulse length of 200 nanoseconds can be made by examining Figures 12 through 15. It is apparent that lasing also starts sooner for the 4000K system.

The effects of pulse length are as expected for total energy. The longer the excitation pulse





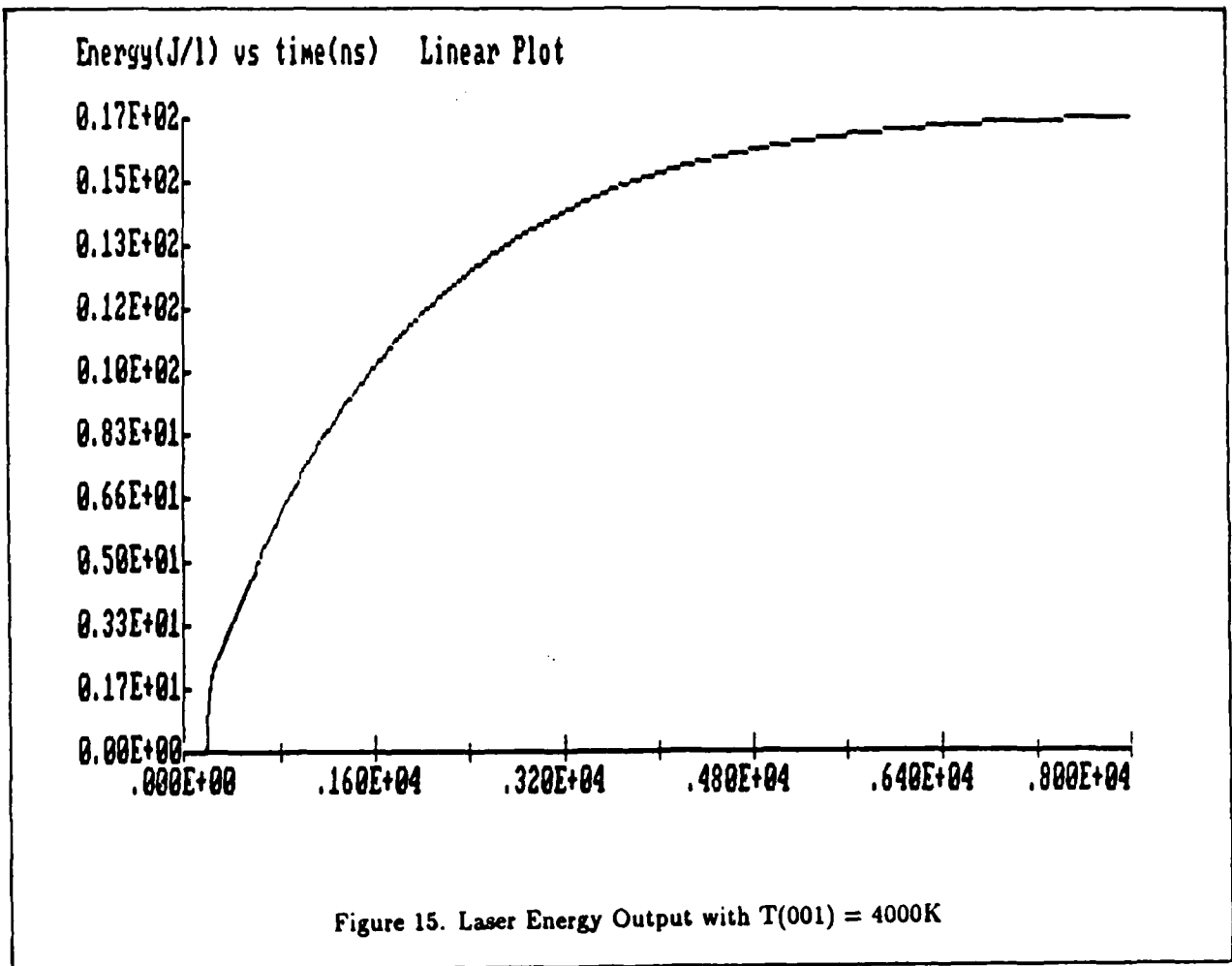


Table 10. Laser Output Versus E/N

E/N (Td)	Peak Power (MW/l)	Total Energy density (J/l)
5	14	3.6
10	64	13
20	75	25
30	78	28
40	75	26

lasts, the more electrical energy will be converted to optical energy. That peak power is also increased when pulse length is increased is probably a result of gain switching, although the difference seen here is very small.

The value of E/N is seen in equation (39) and (54a) as having a direct influence on the energy gained by the electrons from the field. We would expect from the cross section data that a higher E/N would make available more energy for transfer from the electrons to N<sub>2</sub>. This can also be seen if the value of N is held constant and E is increased. A higher value for the applied field means a greater accelerating force is applied to the electrons. These more energetic electrons can then transfer more energy to the other species. Alternatively, if we hold E constant and lower N, the mean free path between collisions will be increased. This means that the electrons will accelerate to a higher velocity before a collision occurs than in the case of the higher N value. Thus the electrons will again be more energetic at the time of collision. Table 10 shows the effects on laser output of varying E/N. The computer runs of CO<sub>2</sub>OSC are for a pulse length of 200 nanoseconds using 100 bins. All temperatures are at 300 K.

The results presented in Table 10 follow in general the expected trend. However, the last set of data for E/N of 40 Td appear to show a reversal of the previous trend. Examination of Figure 16, the plot of the normalized electron number density distribution for this run, shows the problem. The distribution function is showing signs of divergence. Switching to a smaller bin width would not help in this case, since the resulting calculation of the distribution would not enable a full description of the entire energy range covered by the distribution. This may also help explain the

Electron No. Density Number of bins = 100  $d_e = 0.0400\text{eV}$   $h = 10.000\text{E}-08$   
Linear Plot Temp = 300 Pressure = 1 atm  $E/N = 4.0000\text{-}16 \text{ V cm}^2$

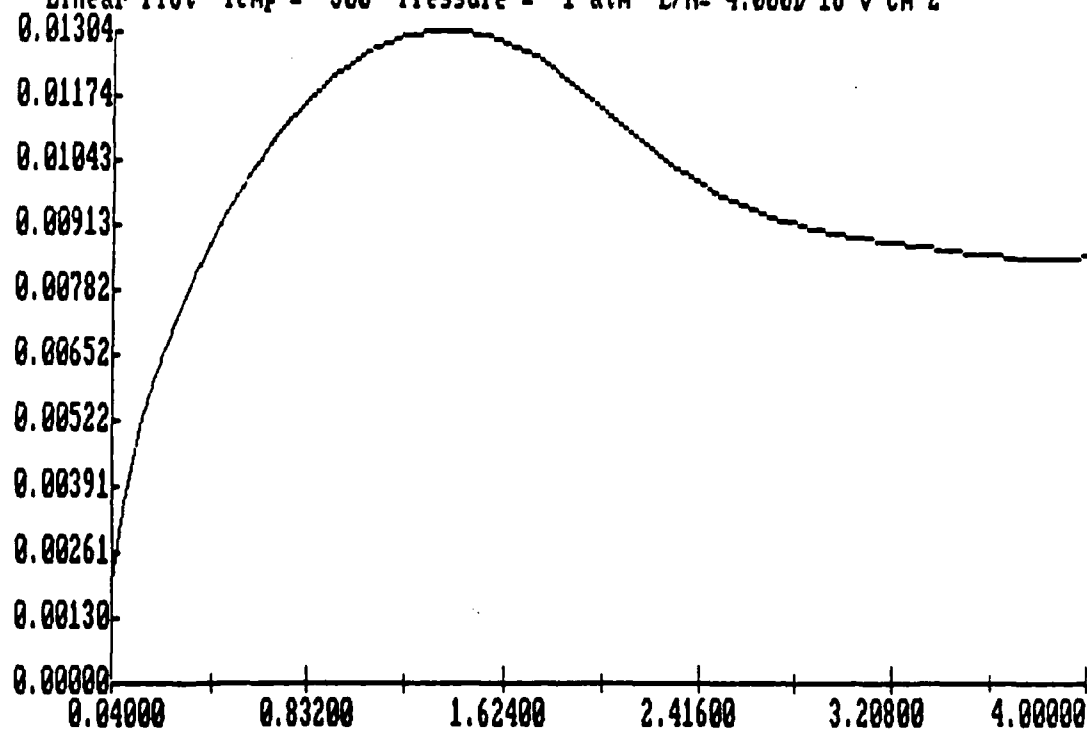


Figure 16. Electron Number Density for Laser Mixture at 40 Td

Table 11. Laser Output Versus Current Density		
Current Density (amps/cm <sup>2</sup> )	Peak Power (MW/l)	Total Energy Density (J/l)
100	76	17
200	110	34

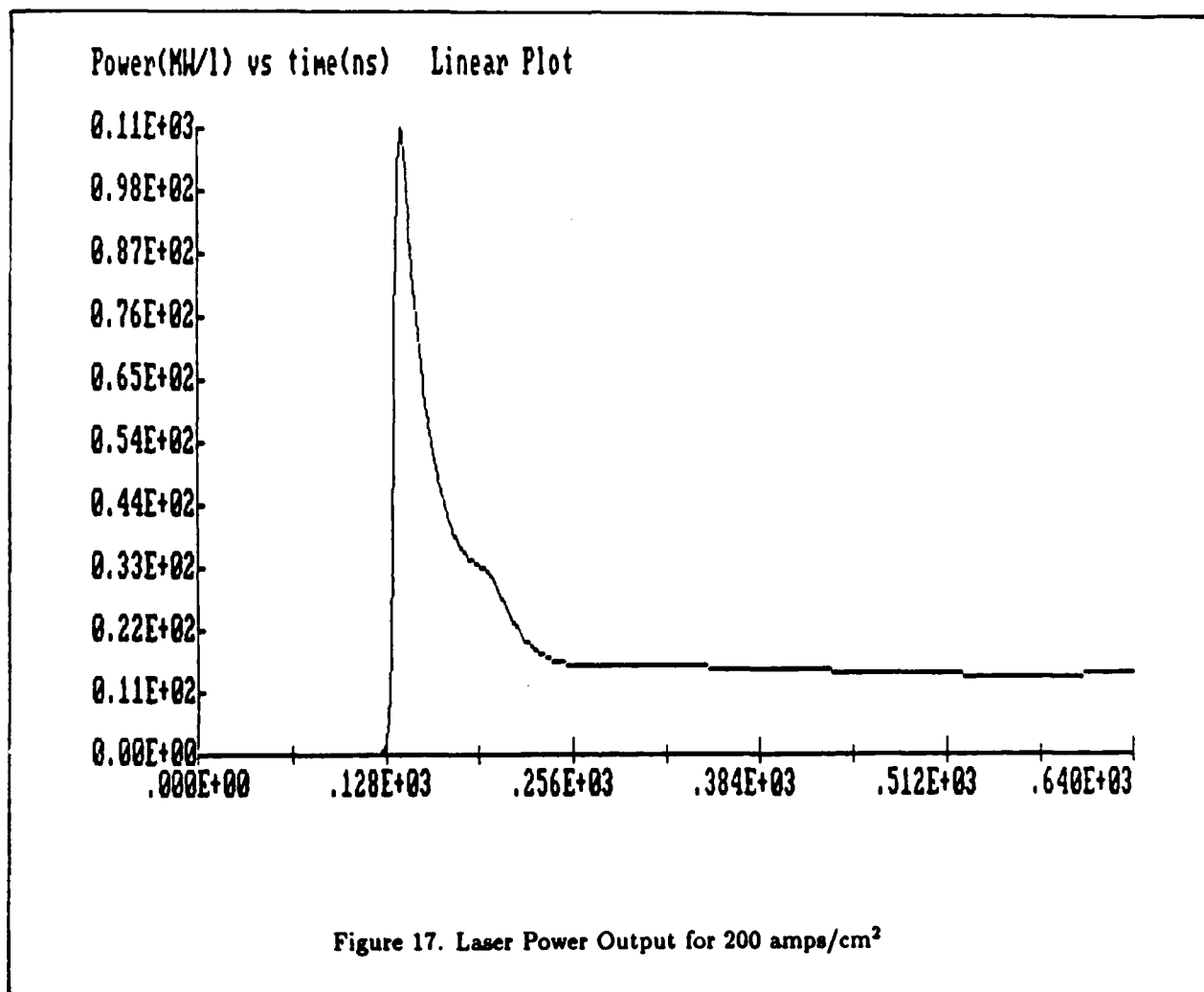
observed differences between the Elliott and CO<sub>2</sub>OSC data of Table 8. Therefore we conclude that the 40 Td run is unreliable.

The final Boltz parameter examined here is current density. It can be seen from section V.A. that plasma tube current density is the circuit parameter used to determine the electron number density. This is described in the equation

$$N_e = (\text{current density}) / (\text{drift velocity} \times \text{electron charge})$$

Thus the electron number density is directly proportional to the current density. We would therefore expect that as the current density is increased, more electrons will be present. If more electrons are now available for collisions, then there should be a greater flow of energy from the applied field to the excited states of the neutral species. This in turn leads to the expectation of greater laser output power and energy when current is increased, even when the other parameters (including E/N) are held constant. Table 11 shows the results from CO<sub>2</sub>OSC for two runs where only the tube current density was changed. The first run had a current density of 100 amps/cm<sup>2</sup> and the second 200 amps/cm<sup>2</sup>. Both used 100 bins with a bin width of .055 eV and E/N of 10 Td. All other parameters were left set at the nominal values.

The expected behavior is seen. More of the energy is transferred from the applied field to the electrons and then to the excited states of the CO<sub>2</sub> and N<sub>2</sub> molecules when plasma tube current density is increased. Figures 14 and 17 show the peak power plots for the 100 and 200 amps/cm<sup>2</sup> cases respectively. Aside from the already stated difference in peak power, two other differences stand out. First, the system with the higher current density begins lasing sooner. Since more



electrons are available for impact excitation of the neutral species, the inversion needed to exceed threshold for lasing can occur more quickly. The second difference seen is in the backside of the main power peak. The higher current density system has its descent slowed near the bottom, resulting in a somewhat wider pulse than the lower current density system. Again, this is a result of the greater level of pumping that takes place when there are more electrons available for impact excitation.

The nominal values for the Boltz portion of CO2OSC and suggested operating limitations are based in part on the results presented above, and in part on the work of others. Boltz and CO2OSC have no internal means for checking on the reasonableness of the values a user might choose for the parameters. It is the responsibility of the user to make sure that all quantities are realistic and not in conflict with each other.

The nominal value for E/N is 10 Td. This value was tested thoroughly for the single gas (N<sub>2</sub>), and the mixture CO<sub>2</sub>, N<sub>2</sub> and He in the ratio .1/.1/.8. No sign of divergence or instability was observed. The recommended range for the gas mixture is 5 to 30 Td. Above 30 Td divergence appears to seriously affect the output for the gas mixture. It is not known if this is true for all other ratios. The vibrational temperatures of CO<sub>2</sub> modes (100) and (010) are set at the corresponding ambient temperature of 300 K. The effective vibrational temperature of mode (001) is set at 4000 K. Smith (21:2042) has indicated that small signal gain in a CO<sub>2</sub> laser was a maximum when this value was used in computer simulations he performed. The number of bins is set at 30. The results obtained indicate that this is the best compromise between speed and accuracy. An energy range of 4 eV appears to be adequate for the range of E/N specified above. This means that the bin width must be set to .1333 eV. The nominal value for current density comes from an article on V-I characteristics of CO<sub>2</sub> lasers by Denes (22:131).

## *V. Conclusions*

The subprogram Boltz, written to provide pumping terms for the CO<sub>2</sub> laser design code CO<sub>2</sub>OSC, has been evaluated against experimental data and the calculations of similar programs. The transport coefficients which Boltz generates for use by CO<sub>2</sub>OSC appear to be very good, when used within the guidelines established in the previous section. As noted, the user of CO<sub>2</sub>OSC must always be alert for the signs of errors in the calculation of the normalized electron number density distribution. Since Boltz has not been tested for all possible ratios of CO<sub>2</sub>, N<sub>2</sub> and He it is impossible to guarantee that the results presented here can be arbitrarily extended to other cases. Each individual set of parameters used must be carefully evaluated to prevent an erroneous set of data from causing a faulty conclusion.

The user of CO<sub>2</sub>OSC must decide the degree of accuracy with which the final laser output calculations will be performed. A more accurate answer requires more energy bins, which in turn means longer run times. Two options allow for a reduction in total computer time. First, initial runs could be made with a smaller number of bins in order to observe trends in the laser output power and density. As undesirable combinations are eliminated, a more accurate computation could then be made. Second, if those parameters affecting electron kinetics can be fixed and run once, then parameter y7 can be set to the "yes" option. This would allow subsequent computer runs to use the most recently calculated pumping terms, avoiding the need for CO<sub>2</sub>OSC to call Boltz. These two options can offer considerable savings in time.

Finally, we have seen the effects on laser output power and energy as the Boltz related parameters were varied. In general, results matched expectations in so far as trends were concerned. No experimental data and no calculations from other programs were available for comparison with the output from CO<sub>2</sub>OSC. In the case where an expected result was not achieved, the underlying problem appeared to stem from failure of the calculated distribution to go to zero at higher energies.

A set of nominal parameters has been programmed into CO2OSC. When used within the guidelines stated previously, computational inaccuracies should be no greater than those listed here.

## VI. Recommendations

Certain projects could be undertaken which might enhance the speed and accuracy of Boltz and CO2OSC. These projects concern the numerical methods used by Boltz and the set up of CO2OSC. Neither Boltz nor CO2OSC model all of the physical processes at this point which might be of interest.

Among the methods tested, L-U decomposition offered the best results for speed and accuracy in solving for the inverse of the (I-hC) matrix. However, the routine used is general in its applicability to all square matrices. A Boltz routine L-U decomposition may offer a substantial improvement in speed. At present, the L-U decomposition routine may not be making best use of the fact that (I-hC) is a sparse, banded matrix. Such an optimized algorithm may already exist, or the present one modified to as to take advantage of the special characteristics of the (I-hC) matrix.

The numerical solution to the Boltzmann equation proposed by Rockwood and upon which Boltz is based, might also warrant revision. A technique using an adaptive energy grid could offer a new and faster approach. This method offers the possibility of getting higher resolution in the electron energy distribution in areas where most needed, with a minimum of mesh points overall. Such a technique has been discussed by Nickel (20).

Another approach which could be used to speed up the Boltz portion of CO2OSC would be to use an interpolation table. If all but one parameter could be fixed, a lookup table could be generated over the dynamic range of interest. Thus, CO2OSC could consult the lookup table instead of calling Boltz.

Boltz itself does not take into account all of the physical processes which a particular user might wish to model. The effects of ionization for example might be interesting to include. While it would not be prudent to try to operate a laser with a plasma that is fully ionized, the effects of various levels of ionization could be modeled.

CO2OSC could also be improved in its ability to model the effects of circuit parameters. At present, only plasma tube current density is included. External excitation by electron beam would be interesting to model. A technique which would couple the external driving circuit equations to the plasma kinetic equations and solve them in a self consistent manner is one possible approach.

Finally, more work can be done with the program just as it stands. The results presented here only explore the behavior of CO2OSC over a small dynamic range of the input parameters. A more exhaustive set of test runs should be conducted over time so as to identify and document those regions where instabilities exist.

## Appendix A. Program Boltz

```

DEFSNG I-J
SUB Boltz (kc%, de, pres, d2, d1, d3, T, VT(), EM, jd!, Wa,
Wb,Wc)
DEFDBL A-Z
DEFSNG D-E, H, J-K, P-Q, T, V-W
'***** Program BOLTZ31.BAS - computation of n(u) and
Vd*****
'***** Elastic, inelastic and superelastic terms for N2,CO2
andHe *****
'***** Incorporates N2DATA, CO2DATA and HEDATA
files*****
'***** User defined Temp, Pressure &
E/N*****

      'NOTE!! - arrays are dynamic, QuickBasic
must bebooted
      'using qb/ah in order to exceed 64K memory
limit.

PRINT ""
PRINT "Boltzmann calculation in progress."
DIM VB(15)
STATIC event
IF event = 1 GOTO label4:      'test to see if this routine
has already been
event = 1                      'run. If so, do not re-load
data.
STATIC eV(), QNj(), eVm1(), eVm2(), eVm3(), QNm1(), QNm2(),
QNm3()
STATIC TempeV(), TempQj(), Tempy(), jin(), K(), ujs(),
Ns(),mjs%()
STATIC jm1, jm2, jm3
'***** Load Cross Section
Data*****
OPEN "N2DATA" FOR INPUT AS #1
OPEN "CO2DATA" FOR INPUT AS #2
OPEN "HEDATA" FOR INPUT AS #3
DIM eV(2, 10, 50), QNj(2, 10, 50), eVm1(70), eVm2(70),
eVm3(70)
DIM QNm1(70), QNm2(70), QNm3(70)          'see
definitions below
DIM TempeV(50), TempQj(50), Tempy(50)    'for use with linear
interpolator
DIM jin(3), K(3, 15), ujs(3, 15), Ns(3), mjs%(3, 15)
'CLS

'***** Read N2 inelastic cross section

```

```

data*****
      'Note: data is in cm-2, and is converted below to
m-2.
INPUT #1, jin(1)          'jin = # of inelastic processes
(excited states)
FOR j = 1 TO jin(1)      'now, step through the excited
states
      INPUT #1, K(1, j), T$, ujs(1, j)
                        'k(1,J) = # of cross sections
for this state
                        'T$ - name of the gas and the
excited state
                        'ujs - inelastic threshold
energy of state j
      FOR I = 1 TO K(1, j)  'step through the energy
levels of state j.
      INPUT #1, eV(1, j, I), QNj(1, j, I) 'eV - energy
inelectron volts for
      QNj(1, j, I) = QNj(1, j, I) / 10000# 'energy I
(orbin) of state j.
      NEXT I                'QNj - inelastic
cross section for
NEXT j                      'energy I (or bin)
of state j.

```

```

'***** Read N2 Momentum Transfer Cross Section Data
*****
      'Note: data is in cm-2, and is converted below to
m-2.
INPUT #1, jm1, MT$        'jm = # of cross sections, MT$ -
name of the gas
FOR j = 1 TO jm1          'step through the energy levels
      INPUT #1, eVm1(j), QNm1(j) 'eVm - energy in
electron volts
      QNm1(j) = QNm1(j) / 10000# 'QNm - momentum
transfer cross section
NEXT j

```

```

'***** Read CO2 inelastic cross section
data*****
      'Note: data is in cm-2, and is converted below to
m-2.
INPUT #2, jin(2)          'jin = # of inelastic processes
(excited states)
FOR j = 1 TO jin(2)      'now, step through the excited
states
      INPUT #2, K(2, j), T$, ujs(2, j)
                        'k(2,J) = # of cross sections
for this state
                        'T$ - name of the gas and the
excited state

```

```

                                'ujs - inelastic threshold
energy of state j
  FOR I = 1 TO K(2, j)      'step through the energy
levels of state j.
    INPUT #2, eV(2, j, I), QWj(2, j, I) 'eV - energy
inelectron volts for
    QWj(2, j, I) = QWj(2, j, I) / 10000# 'energy I
(orbin) of state j.
    NEXT I                  'QWj - inelastic
cross section for
NEXT j                      'energy I (or bin)
of state j.

```

'\*\*\*\*\* Read CO2 Momentum Transfer Cross Section

Data\*\*\*\*\*

'Note: data is in cm<sup>-2</sup>, and is converted below to  
m<sup>-2</sup>.

```

INPUT #2, jm2, MT$          'jm = # of cross sections, MT$ -
name of the gas
FOR j = 1 TO jm2            'step through the energy levels
  INPUT #2, eVm2(j), QNm2(j) 'eVm - energy in
electron volts
  QNm2(j) = QNm2(j) / 10000# 'QNm - momentum
transfer cross section
NEXT j

```

'\*\*\*\*\* Read He Momentum Transfer Cross Section

Data\*\*\*\*\*

'Note: data is in cm<sup>-2</sup>, and is converted below to  
m<sup>-2</sup>.

```

INPUT #3, jm3, MT$          'jm = # of cross sections, MT$ -
name of the gas
FOR j = 1 TO jm3            'step through the energy levels
  INPUT #3, eVm3(j), QNm3(j) 'eVm - energy in
electron volts
  QNm3(j) = QNm3(j) / 10000# 'QNm - momentum
transfer cross section
NEXT j

```

'\*\*\*\*\*

label4:

```

FOR I = 1 TO 3              'set Boltz vibrational temps to
input
values
  VB(I) = VT(I)
NEXT I
FOR I = 4 TO jin(2)        'set the remaining modes equal to
ambient temp

```

```

      VB(I) = T
NEXT I

REDIM abar(kc%), bbar(kc% + 1)    'for use in energy balance
& Vd
REDIM A(kc%), b(kc% + 1), RTjs(kc%), Rjs(2, 9, kc%)
REDIM RSjs(2, 9, kc%), RTSjs(kc%)    'Superelastics
REDIM c(kc%, kc%), NO(kc%)          'C= constants
REDIM NO1(kc%)                      'NO = output energy densities
REDIM INdx(kc%), vv(kc%), yb(kc%, kc%) 'used in back-sub.
REDIM PNO(kc%)                      'temporary single prec. storage of NO
for
plotting

'*** Defining of Input Parameters and Constants and Terms in
S.I. units***
CONST Pi = 3.141592654#
conv = .00001#    'convergence criteria for iteration loop
to end
' j = excited state, s = species
' ek = value of right hand edge of energy cell
'Qms - momentum transfer cross section for elastic
collisions of
      'electrons (of energy ekp) with molecules of species
"s" (m-2)
'Qjs - inelastic cross section for this process (m-2)
'ujs - energy loss in e-volts for the j-th inelastic process
in species s
'T molecular gas temperature (K)
'N = total molecular number density in 1/m3
'Ns1 - molecular number density of species 1
'ds1 - number density of molecules of species 1/N -delta
sub s
'E = applied field in V/m
'EN = E/N - input as Volts cm-2, then converted to Volts m-2
'de = width of cells along the energy axis - delta "e"
(electron volts)
'P= pressure - input as atmospheres, then converted to
Pascals
q = (1.6022D-19) ^ 2    'square of electron charge - (e-2)
ec = 1.6022D-19    'electron charge
m = 9.109399999999999D-31 'electron mass
KB = 8.617400000000001D-05 'Boltzmann constant in eV/K
kg = 1.3807D-23 'J/K - S.I. Boltzmann constant in J/K
h = .0000001# 'time step
Ms1 = 4.6518D-26    'mass of molecular N2 in kg
Ms2 = 7.3045D-26    'mass of molecular CO2 in kg
Ms3 = 6.6463D-27    'mass of He atom in kg

P = pres * 110133#
N = P / (kg * T)    'gas density in m-3

```

```

EOM = EM / 10000#           'convert to V m-2
E = EOM * N                 'convert to V/m
d1 = d1 / 100               'convert to fraction amount of
each gas- N2
d2 = d2 / 100               'CO2
d3 = d3 / 100               'He
Ns(1) = N * d1              'total number density of each
species
Ns(2) = N * d2
Ns(3) = N * d3

'***** mjs% = nearest integer to ujs/de *****
'CLS
FOR j = 1 TO 2              'each species
  FOR I = 1 TO jin(j)      'each of the excited
states
    mjs%(j, I) = CINT(ujs(j, I) / de)
  NEXT I
NEXT j

' ***** Calculate Elements of a, b, and Rjs
Matrices*****
'***** Calculate Rjs(J,K) matrix elements and the RTjs(K)
matrix elements *
'***** RTjs(K) - sum over all processes
(J).*****
FOR K = 1 TO 2              'species
  FOR I = 1 TO jin(K)      'number of excited states
in a species
    FOR j = 1 TO K(K, I)   '# of data points
available for interp.
      TempeV(j) = eV(K, I, j)
      TempQj(j) = QNj(K, I, j)
    NEXT j
    FOR j = 1 TO kc%       'energy bin
      ek = j * de
      CALL Interp(TempeV(), TempQj(), K(K, I), ek,
Qjo)
      Rjs(K, I, j) = Qjo * SQR(2 * ek * ec / m) *
Ns(K)
      RTjs(j) = RTjs(j) + Qjo * SQR(2 * ek * ec /
m) * Ns(K)
      CALL Interp(TempeV(), TempQj(), K(K, I), ek +
uj(K, I), QSjo)
      RSjs(K, I, j) = ((ek + ujs(K, I)) / ek) * QSjo *
SQR(2 * ek * ec / m)
      IF K = 1 THEN        'selectively apply the correct
vibrational temp
        RSjs(K, I, j) = RSjs(K, I, j) * Ns(K) *
EXP(-uj(K, I) / (T))
      END IF

```

```

      IF K = 2 THEN
        RSjs(K, I, j) = RSjs(K, I, j) * Ms(K)
        RSjs(K, I, j) = RSjs(K, I, j) * EXP(-ujs(K, I) /
(KB* VB(I)))
      END IF
      RTSjs(j) = RTSjs(j) + RSjs(K, I, j)

      *** Note: Rjs and RTjs include multiplication by Ms **
    NEXT j
  NEXT I
NEXT K
FOR I = 1 TO kc%
  ek = I * de
  CALL Interp(eVm1(), QWm1(), jm1, ek, Qms1)
  CALL Interp(eVm2(), QWm2(), jm2, ek, Qms2)
  CALL Interp(eVm3(), QWm3(), jm3, ek, Qms3)
  nup = (SQR(2 * ec * ek / m)) * (d1 * Qms1 + d2 * Qms2 +
d3 * Qms3)
  nub = (d1 * Qms1 / Ms1) + (d2 * Qms2 / Ms2) + (d3 *
Qms3 / Ms3)
  nub = nub * (SQR(2 * ec * ek / m)) * 2 * m * N
  A(I) = (2 * N * ec / (3 * m)) * ((E / N) ^ 2) * (1 /
nup) * (de ^ -2)
  A(I) = A(I) * (ek + (de / 4)) + (nub / (2 * de)) * (KB
* T / 2)
  A(I) = A(I) + (nub / (2 * de)) * (-ek + (2 * KB * T /
de) * ek)
  b(I + 1) = (2 * N * ec / (3 * m)) * ((E / N) ^ 2) * (1
/nup) * (de ^ -2)
  b(I + 1) = b(I + 1) * (ek - (de / 4)) - (nub / (2 *
de)) * (KB * T / 2)
  b(I + 1) = b(I + 1) + (ek + (2 * KB * T / de) * ek) *
(nub / (2 * de))
NEXT I

***** Load (I-hc)-Matrix *****
b(1) = 0          'boundary condition
A(kc%) = 0        'boundary condition
c(1, 2) = -h * b(2)
c(1, 1) = 1 + h * (A(1) + b(1) + RTjs(1) + RTSjs(1))
c(kc%, kc% - 1) = -h * A(kc% - 1)
c(kc%, kc%) = 1 + h * (A(kc%) + b(kc%) + RTjs(kc%) +
RTSjs(kc%))
c(kc%, kc%) = c(kc%, kc%) + h * RTSjs(kc%)
FOR i = 2 TO kc% - 1
  c(i, i - 1) = -h * A(i - 1)
  c(i, i + 1) = -h * b(i + 1)
  c(i, i) = 1 + h * (A(i) + b(i) + RTjs(i) + RTSjs(i))
NEXT I
FOR K = 1 TO 2          'species
  FOR I = 1 TO kc%      'energy bin
    FOR j = 1 TO jin(K) 'excited state

```

```

        IF I + mjs%(K, j) <= kc% THEN
            cim = -h * Rjs(K, j, I + mjs%(K, j))
            c(I, I + mjs%(K, j)) = cim + c(I, I +
mjs%(K, j))
        END IF
        IF I - mjs%(K, j) >= 1 THEN
            cim1 = -h * RSjs(K, j, I - mjs%(K, j))
            c(I, I - mjs%(K, j)) = cim1 + c(I, I
-mjs%(K, j))
        END IF
    NEXT j
NEXT I
NEXT K
PRINT ""
PRINT "      (I-h*c) Matrix Loaded"

```

```

'***** Calculate the Inverse of
(I-hc)*****
'***** LU Decomposition Matrix Inversion
Algorithm*****
'***** See Numerical Recipes - page
38*****

```

```

'***** Perform LU
Decomposition*****
CONST Tiny = 1E-20
D = 1
FOR I = 1 TO kc%
    Aamax = 0
    FOR j = 1 TO kc%
        IF ABS(c(I, j)) > Aamax THEN
            Aamax = ABS(c(I, j))
        END IF
    NEXT j
    IF Aamax = 0 THEN
        PRINT "Singular Matrix"
        END
    END IF
    vv(I) = 1 / Aamax
NEXT I
FOR j = 1 TO kc%
    FOR I = 1 TO j - 1
        sum = c(I, j)
        FOR K = 1 TO I - 1
            sum = sum - c(I, K) * c(K, j)
        NEXT K
        c(I, j) = sum
    NEXT I
    Aamax = 0
    FOR I = j TO kc%

```

```

        sum = c(I, j)
        FOR K = 1 TO j - 1
            sum = sum - c(I, K) * c(K, j)
        NEXT K
        c(I, j) = sum
        dum = vv(I) * ABS(sum)
        IF dum >= Aamax THEN
            Imax = I
            Aamax = dum
        END IF
    NEXT I
    IF j <> Imax THEN
        FOR K = 1 TO kc%
            dum = c(Imax, K)
            c(Imax, K) = c(j, K)
            c(j, K) = dum
        NEXT K
        D = -D
        vv(Imax) = vv(j)
    END IF
    INdx(j) = Imax
    IF c(j, j) = 0 THEN
        c(j, j) = Tiny
    END IF
    IF j <> kc% THEN
        dum = 1 / c(j, j)
        FOR I = j + 1 TO kc%
            c(I, j) = c(I, j) * dum
        NEXT I
    END IF
NEXT j

'***** Back
Substitution*****
FOR I = 1 TO kc%
    'load identity matrix
    FOR j = 1 TO kc%
        yb(I, j) = 0
    NEXT j
    yb(I, I) = 1
NEXT I

FOR j = 1 TO kc%
    II = 0
    FOR I = 1 TO kc%
        LL = INdx(I)
        sum = yb(LL, j)
        yb(LL, j) = yb(I, j)
        IF II <> 0 THEN
            FOR J1 = II TO I - 1
                sum = sum - c(I, J1) * yb(J1, j)
            NEXT J1

```

```

        ELSEIF sum <> 0 THEN
            II = I
        END IF
        yb(I, j) = sum
    NEXT I
    FOR I = kc% TO 1 STEP -1
        sum = yb(I, j)
        IF I < kc% THEN
            FOR J1 = I + 1 TO kc%
                sum = sum - c(I, J1) * yb(J1, j)
            NEXT J1
        END IF
        yb(I, j) = sum / c(I, I)
    NEXT I
NEXT j
PRINT "          Inverse of (I-hc) computed"

'***** Load initial values for NO
matrix*****
FOR I = 1 TO kc%
    NO(I) = 1
NEXT I

'***** Calculate NO matrix through
iteration*****
FOR K = 1 TO 300          'sets the maximum number of
iterations
    FOR I = 1 TO kc%      'calculate the new electron number
density
        sum = 0
        FOR j = 1 TO kc%
            sum = sum + (yb(I, j) * NO(j))
        NEXT j
        NO1(I) = sum
        'PRINT "I = "; I; " NO(I) "; NO(I); " NO1(I)";
    NO1(I)
    NEXT I
    sum = 0          'calculate the normalization
factors for NO & NO1
    sum1 = 0
    FOR I = 1 TO kc%
        sum = sum + (NO(I))
        sum1 = sum1 + (NO1(I))
    NEXT I
    cmax = 0          'maximum difference value for a
particular iteration
    FOR I = 1 TO kc%      'Determine convergence - DIFF
values
        norm = NO(I) / sum
        norm1 = NO1(I) / sum1
        diff = ABS(norm - norm1) / norm
    
```

```

        IF diff > cmax THEN
            cmax = diff
        END IF
        NO(I) = NO1(I)      'Set up NO values for next K
iteration
        NEXT I
        iter = K           'Used below to print out number
of iterations
        IF cmax < conv THEN 'Test for convergence
            K = 300
        END IF
    NEXT K
    FOR I = 1 TO kc%
        NO(I) = NO(I) / sum1 'normalization
        PNO(I) = NO(I)      'storage for plotting below, to
prevent parameter
                                'type mismatch
    NEXT I

    '***** Plot the normalized electron number density
distribution *****
    BEEP
    CALL Plot1(de, kc%, PNO(), h, T, P, EON) 'plot routine
just for no. dens.

    '***** Calculate a-bar and b-bar for drift velocity
calculations *****
    FOR I = 1 TO kc%
        ek = I * de
        CALL Interp(eVm1(), QNm1(), jm1, ek, Qms1)
        CALL Interp(eVm2(), QNm2(), jm2, ek, Qms2)
        CALL Interp(eVm3(), QNm3(), jm3, ek, Qms3)
        nup = (SQR(2 * ec * ek / m)) * (d1 * Qms1 + d2 * Qms2 +
d3 * Qms3)
        nub = (d1 * Qms1 / Ms1) + (d2 * Qms2 / Ms2) + (d3 *
Qms3 / Ms3)
        nub = nub * (SQR(2 * ec * ek / m)) * 2 * m * H
        abar(I) = (2 * H * ec / (3 * m)) * ((E / H) ^ 2) * (1 /
nup) * (de ^ -2)
        abar(I) = abar(I) * (ek + (de / 4))

        bbar(I + 1) = (2 * H * ec / (3 * m)) * ((E / H) ^ 2) * (1
/nup) * (de ^ -2)
        bbar(I + 1) = bbar(I + 1) * (ek - (de / 4))
    NEXT I

    '***** Calculate Drift Velocities
*****
    CLS 0
    PRINT "Loop ended with iteration "; iter;
    PRINT USING " Difference = ###.###^---"; cmax

```

```

PRINT "Number of bins ="; kc%; " de=";
PRINT USING "##.###"; de;
PRINT "eV ";
PRINT USING "  N2/CO2/He = ##./##./##"; d1; d2; d3
Vd = 0
ebar = 0
FOR I = 1 TO kc%          'Calculate numerical drift
velocity
  Vd = Vd + (abar(I) - bbar(I)) * NO(I)
  ebar = ebar + (NO(I) * (I * de))
NEXT I
Vd = Vd * de / E
PRINT USING "T =### T(100) =###"; T; VB(2);
PRINT USING "  T(010) =### T(001) =###"; VB(1); VB(3)
PRINT USING "E/W = ##.###-----"; (E / W) * 100000;
PRINT USING "Vcm^2 Numerical Drift Velocity =
##.###-----"; Vd/.01;
PRINT "cm/s"
PRINT USING "          Average electron energy =
##.###-----"; ebar;
PRINT "eV"

'***** Min and Max Values of the Distribution
*****
Max = NO(1)          'Determine range of values
for y-axis
Min = NO(1)
FOR I = 1 TO kc%
  IF NO(I) > Max THEN
    Max = NO(I)
  END IF
  IF NO(I) < Min THEN
    Min = NO(I)
  END IF
NEXT I
PRINT USING "Max value of distribution = ##.###-----"; Max;
PRINT USING "  Min value of distribution = ##.###-----"; Min

'***** Excitation Rate Calculations - From Rockwood and
Greene, eq.(18) **
          '***** Nitrogen *****
PRINT ""
PRINT "          Excitation Rates for Nitrogen"
TOTN2RATE = 0
FOR I = 1 TO jin(1)          'number of excited states in
species
  FOR j = 1 TO K(1, I)      '* of data points available
for interp.
    TempeV(j) = eV(1, I, j)
    TempQj(j) = QNj(1, I, j)
  NEXT j

```

```

N2RATE = 0
FOR j = 1 TO kc%      'energy bin
    ek = j * de
    CALL Interp(TempeV(), TempQj(), K(1, I), ek, Qjo)
    N2RATE = N2RATE + Qjo * SQR(2 * ek * ec / m) *
NO(j)
NEXT j
PRINT "v = "; I; " Rate = ";
PRINT USING "##.####^----"; N2RATE * 1000000#;
'conversion to cm3
PRINT " cm3/sec"
TOTN2RATE = TOTN2RATE + N2RATE * 1000000#
TOTN2R = TOTN2R + (I * N2RATE)      'Rate calculation
for term Wc
NEXT I
PRINT USING "Total excitation rate of all levels =
##.###^----";
TOTN2RATE;
PRINT " cm3/sec"

          '***** CO2 *****
'Vibration - (010)
FOR j = 1 TO K(2, 1)
    TempeV(j) = eV(2, 1, j)
    TempQj(j) = QWj(2, 1, j)
NEXT j
CO2R2 = 0
FOR j = 1 TO kc%
    ek = j * de
    CALL Interp(TempeV(), TempQj(), K(2, 1), ek, Qjo)
    CO2R2 = CO2R2 + Qjo * SQR(2 * ek * ec / m) * NO(j)
NEXT j
'Vibration - (100)
FOR j = 1 TO K(2, 2)
    TempeV(j) = eV(2, 2, j)
    TempQj(j) = QWj(2, 2, j)
NEXT j
CO2R1 = 0
FOR j = 1 TO kc%
    ek = j * de
    CALL Interp(TempeV(), TempQj(), K(2, 2), ek, Qjo)
    CO2R1 = CO2R1 + Qjo * SQR(2 * ek * ec / m) * NO(j)
NEXT j
'Vibration - (001)
FOR j = 1 TO K(2, 3)
    TempeV(j) = eV(2, 3, j)
    TempQj(j) = QWj(2, 3, j)
NEXT j
CO2R3 = 0
FOR j = 1 TO kc%
    ek = j * de

```

```

      CALL Interp(TempeV(), TempQj(), K(2, 3), ek, Qjo)
      CO2R3 = CO2R3 + Qjo + SQR(2 * ek * ec / m) * NO(j)
NEXT j
PRINT "                      Excitation rates for CO2"
PRINT USING "Excitation rate for (100) = ###.###^-----"; CO2R1
*1000000#;
PRINT "cm^3/sec"
PRINT USING "Excitation rate for (010) = ###.###^-----"; CO2R2
*1000000#;
PRINT "cm^3/sec"
PRINT USING "Excitation rate for (001) = ###.###^-----"; CO2R3
*1000000#;
PRINT "cm^3/sec"

'***** Calculation of Pumping Terms
*****
CD = jd! * 10000!      'Conversion of current density to A/m^2
Ne = CD / (Vd * ec)    'Electron number density in
1/(m^3)
Wa = Ne * (d2 * N) * CO2R3      '1/(s*m^3)
Wb = Ne * (d2 * N) * CO2R1      '1/(s*m^3)
Wc = Ne * (d1 * N) * TOTW2R      '1/(s*m^3)

'***** Energy balance check
*****
Egain = 0
Eelastic = 0
Eincl = 0
FOR I = 1 TO kc%
  Egain = Egain + (abar(I) - bbar(I)) * NO(I) * de
  Eelastic = Eelastic - (A(I) - abar(I) - b(I) + bbar(I))
* NO(I) * de
  FOR K = 1 TO 2      'species
    FOR j = 1 TO jin(K)      'excited state
      IF I + mjs%(K, j) <= kc% THEN
        Eim = Rjs(K, j, I + mjs%(K, j))
        Eim = Eim * NO(I + mjs%(K, j)) * mjs%(K,
j) * de
        Eincl = Eincl + Eim
      END IF
      IF I - mjs%(K, j) >= 1 THEN
        Eim1 = RSjs(K, j, I - mjs%(K, j))
        Eim1 = Eim1 * NO(I - mjs%(K, j)) *
mjs%(K, j) * de
        Eincl = Eincl - Eim1
      END IF
    NEXT j
  NEXT K
NEXT I
PRINT "Energy balance is good to ";
PRINT USING "###.###^-----"; ABS((Egain - Eelastic - Eincl) /

```

```
Egain)
PRINT ""
INPUT "Press Return to Continue", ff$
ERASE abar, bbar, A, b, RTjs, Rjs, RSjs, RTSjs, c, NO, NO1,
INdx, vv, yb, PNO
CLS
END SUB
```

## Appendix B. Program CO2OSC Modifications

```

'***** CO2OSC - CO2 Laser Design Software
*****
DECLARE SUB Plot2 (h1!, h2!, jChStep%, ChTimeStep!, tCav!,
jmax%,
Yquant!(), title$)
DECLARE SUB cgadump (res!)
DECLARE SUB Boltz (kc%, de!, pres!, d2!, d1!, d3!, T!,
VT!(),
EN!, jd!, Wa!, Wb!, Wc!)
DECLARE SUB Interp (x!(), y!(), N!, xin!, yout!)
DECLARE SUB Plot1 (de!, kc%, NO!(), h!, T!, P!, EN!)
DECLARE SUB RungeKutta4 (t0!, y0!(), T!, y!(), h!)
DECLARE FUNCTION f! (I%, T!, y!())
CLEAR , , 1700 'Allows increased stack space for
subprocedures & functions.
'This CO2 pulsed laser model assumes 4-level operation. It
incorporates
' choice of integration time steps to track both the
"pulse" and the
' "tone" portions of the laser pulse.

'***** NOMINAL INPUT PARAMETER LIST *****

Lres = 1: refl = 71.6364: areaSec = .0004
pres = 1: pctCO2 = 10: pctN2 = 10: pctHe = 80
pctH2O = 0: pctH2 = 0: temp = 300
tauPump = 10: tmax = 400: h1 = .1: h2 = 1
pumpEff = .2: tChStep = 30: ChTimeStep = 1: fileStep = 5
timeFileUnits = 0: isotope = 0: resonator = 1: alpha = 3
qSwitch = 0: timeqSwitch = 15: pulseShape = 0: EN = 1E-16

'***** For use with Sub Boltz
*****
DIM VT(15)
VT(2) = 300: VT(1) = 300: VT(3) = 4000: kc% = 30: de =
.1333: jd!
= 100
Prev$ = "n"

inputroutine:
n1 = 1: n2 = 3 'make array size variable so compiler
allocates dynamically
IF ChTimeStep = 0 THEN n3 = tmax / h1 ELSE n3 = tChStep / h1
+
(tmax - tChStep) / h2

```

```

REDIM SHARED Pop(n1 TO n2, n3), Gain(n3), Power(n3),
Energy(n3),
REDIM Pspont4Pi(n3), Espont4Pi(n3) AS SINGLE
REDIM SHARED y0(1 TO 4) AS SINGLE
REDIM SHARED y(1 TO 4) AS SINGLE
DIM SHARED PumpOn AS INTEGER
DIM SHARED Ga, Gb, GcCO2, GcN2, Wa, Wb, Wc, Ws, resonator1,
degenRatio, qSwitch1
tCav = 2 * Lres / cLight / LOG(1 / (refl / 100)) 'cavity
lifetime

COLOR 15, 1
CLS
COLOR 15, 2
PRINT "      INPUT PARAMETERS ";
COLOR 15, 1: PRINT " ";
COLOR 14, 4: PRINT "*** Version "; version$; ", Dave Stone,
"; date1$; " *** "
COLOR 15, 1
PRINT "a. Resonator Length (m) "; Lres
PRINT "b. Output Reflectivity (%) "; refl; " ";
COLOR 15, 2: PRINT " cavity lifetime = ";
PRINT USING "###.##"; tCav * 1E+09;
PRINT " nanosec ": COLOR 15, 1
PRINT "c. Q Switch on-1, off-0 "; qSwitch
PRINT "c1. time to Q Switch "; timeqSwitch; "* tCav"
PRINT "d. Pressure (atm) "; pres; " ";
COLOR 15, 2: PRINT " ** The code's time unit = cavity
lifetime.":
COLOR 15, 1
PRINT "e. Percent CO2 "; pctCO2; " y. E/W
in V cm^-2";
PRINT " "; EM
PRINT "f. Percent H2 "; pctH2; " y1.
Vibrational Temp";
PRINT " of mode (100)"; VT(2)
PRINT "g. Percent HE "; pctHe; " y2.
Vibrational Temp";
PRINT " of mode (010)"; VT(1)
PRINT "h. Percent H2O "; pctH2O; " y3.
Vibrational Temp";
PRINT " of mode (001)"; VT(3)
PRINT "i. Percent H2 "; pctH2; " y4.
Number of energy";
PRINT " bins "; kc%
PRINT "j. Pump Efficiency "; pumpEff; " y5.
Energy cell width";
PRINT " in eV "; de
PRINT "k. Temperature (K) "; temp; " y6. Tube
Current Density";
PRINT " (A/cm^2) "; jd!

```

```

PRINT "1. Pump Pulse Length (t0) "; tauPump; " y7.
Previous Kinetics?";
PRINT " (y/n) "; Prev$
PRINT "11. Pulse Shape 0-rec,1-sin "; pulseShape
PRINT "m. Max Computation Time (t0)"; tmax
PRINT "n. Integration Time Step #1 "; h1
PRINT "p. Integration Time Step #2 "; h2
PRINT "r. Time to Change Time Step "; tChStep
PRINT "s. ? change time step: 0,1 "; ChTimeStep
PRINT "v. C12 - 0 or C13 - 1 "; isotope
PRINT "w. Resonator on-1, off-0 "; resonator
PRINT "x. Unfav loss % per rnd trip"; alpha
COLOR 14, 4
INPUT "Select item to be changed, <q> to quit, or <RETURN>
to run: ", Choice$

SELECT CASE Choice$
CASE "a", "A"
INPUT "Lres = ", Lres
CASE "b", "B"
INPUT "refl = ", refl
CASE "c", "C"
INPUT "qSwitch = ", qSwitch
CASE "c1", "C1"
INPUT "time to Q Switch = ", timeqSwitch
CASE "d", "D"
INPUT "pres = ", pres
CASE "e", "E"
INPUT "pctCO2 = ", pctCO2
CASE "f", "F"
INPUT "pctN2 = ", pctN2
CASE "g", "G"
INPUT "pctHE = ", pctHe
CASE "h", "H"
INPUT "pctH2O = ", pctH2O
CASE "i", "I"
INPUT "pctH2 = ", pctH2
CASE "j", "J"
INPUT "pumpEff = ", pumpEff
CASE "k", "K"
INPUT "temp = ", temp
CASE "l", "L"
INPUT "tauPump = ", tauPump
CASE "l1", "L1"
INPUT "pulse shape = ", pulseShape
CASE "m", "M"
INPUT "tMax = ", tmax
CASE "n", "N"
INPUT "h1 = ", h1
CASE "p", "P"
INPUT "h2 = ", h2

```

```

CASE "r", "R"
    INPUT "tChStep = ", tChStep
CASE "s", "S"
    INPUT "ChTimeStep = ", ChTimeStep
CASE "v", "V"
    INPUT "isotope = ", isotope
CASE "w", "W"
    INPUT "resonator = ", resonator
CASE "x", "X"
    INPUT "alpha = ", alpha
CASE "y", "Y"
    INPUT "E/W in V cm-2 = ", EN
CASE "y1", "Y1"
    INPUT "T(100) in (K) = ", VT(2)
CASE "y2", "Y2"
    INPUT "T(010) = ", VT(1)
CASE "y3", "Y3"
    INPUT "T(001) = ", VT(3)
CASE "y4", "Y4"
    INPUT "Number of cells (kc%) = ", kc%
CASE "y5", "Y5"
    INPUT "cell width de = ", de
CASE "y6", "Y6"
    INPUT "current density (A/cm-2) = ", jd!
CASE "y7", "Y7"
    INPUT "use previous kinetic calculations (y/n) ? ",
Prev$
CASE "q", "Q"
    END
CASE ""
    GOTO MainBody
CASE ELSE
    'Continue
END SELECT

'pump rates and effective spontaneous emission rate

d2 = pctCO2: d1 = pctN2: d3 = pctHe      'For use with Boltz
subprocedure
IF Prev$ = "n" THEN
    CALL Boltz(kc%, de, pres, d2, d1, d3, temp, VT(), EN,
jd!, W1, W2, W3)
    'Wb = Wa
'***** Check to see if Boltz calculation is acceptable
*****
PRINT "Is the Boltzmann calculation acceptable (y/n) ?"
INPUT "    y - continue the program    n - return to
menu", L$
SELECT CASE L$
CASE "n"
    GOTO inputroutine:

```

```

        END SELECT
    END IF
    PRINT "Laser calculation in progress."
    Wa = tCav * W1 / dCrit
    Wb = tCav * W2 / dCrit
    Wc = tCav * W3 / dCrit
    Ws = rSigma / areaSec / Pi
    Wc1 = Wc: Wa1 = Wa: Wb1 = Wb

'Output routines
BEEP
Start:

    COLOR 14, 4
    INPUT "Print again? y or n ", b$
    IF b$ = "y" THEN GOTO Start
    title$ = "Power(MW/l) vs time(ns)"
    CALL Plot2(h1, h2, jChStep, ChTimeStep, tCav, jmax,
Power(), title$)
    title$ = "Energy(J/l) vs time(ns)"
    CALL Plot2(h1, h2, jChStep, ChTimeStep, tCav, jmax,
Energy(), title$)
    INPUT "Make data files? y or n ", c$
    IF c$ = "y" THEN GOTO OutputFiling ELSE GOTO inputroutine

RoutineVariableTimeStep:
    COLOR 14, 4
Finish:
    COLOR 14, 4
    INPUT "Print again? y or n ", b$
    IF b$ = "y" THEN GOTO Start
    title$ = "Power(MW/l) vs time(ns)"
    CALL Plot2(h1, h2, jChStep, ChTimeStep, tCav, jmax,
Power(), title$)
    title$ = "Energy(J/l) vs time(ns)"
    CALL Plot2(h1, h2, jChStep, ChTimeStep, tCav, jmax,
Energy(), title$)

```

## Appendix C. Subprograms

```
DEFINT I-J
DEFSNG A-C, F-G, L-O, R-S, U, X-Z
'***** CGA Screen Dump Utility. Copied from "Microsoft
QuickBASIC for
'Scientists" by James W. Cooper, page 156, Copyright 1988 by
John Wiley, Inc.
SUB cgadump (res)
DIM buf(800) AS INTEGER, I AS INTEGER, j AS INTEGER
DIM col AS INTEGER
esc$ = CHR$(27)
WIDTH "lpt1:", 255
SELECT CASE res
    CASE LORES:
        ctrlchar$ = "K"
        incr = 4
        bytes = 400
    CASE MEDRES:
        ctrlchar$ = "Y"
        incr = 8
        bytes = 800
    CASE HIRES:
        ctrlchar$ = "L"
        incr = 8
        bytes = 800
END SELECT
LPRINT esc$; "3"; CHR$(24)
FOR col = 0 TO 79
    DEF SEG = &HB800
    I = bytes - 1
    FOR j = 0 TO 99
        buf(I) = PEEK(j * 80 + col)
        buf(I - 1) = buf(I)
        IF (res <> LORES) THEN
            buf(I - 2) = buf(I)
            buf(I - 3) = buf(I)
        END IF
        I = I - incr
    NEXT j
    DEF SEG = &HBA00
    I = bytes - 1 - incr / 2
    FOR j = 0 TO 99
        buf(I) = PEEK(j * 80 + col)
        buf(I - 1) = buf(I)
        IF (res <> LORES) THEN
            buf(I - 2) = buf(I)
            buf(I - 3) = buf(I)
        END IF
```

```

        I = I - incr
    NEXT j
    hiByte = bytes \ 256
    loByte = bytes - (hiByte * 256)
    LPRINT esc$; ctrlChar$;
    LPRINT CHR$(loByte); CHR$(hiByte);
    FOR I = 0 TO bytes - 1
        LPRINT CHR$(buf(I));
    NEXT I
    LPRINT
NEXT col
LPRINT CHR$(12);
END SUB

```

```

DEFSNG I
SUB Interp (x(), y(), N, xin, yout)
    'x - input tabulated abscissas
    (known)
    'y - input tabulated ordinates
    (known)
    'n - max index of input arrays
    x & y
    'xin - input abscissa value
    (known)
    'yout - output ordinate value
    (unknown)

    IF xin < x(1) THEN      'Check to see if input x value is in
        yout = 0           'If it isn't, then set yout = 0.
        GOTO label2:
    ELSEIF xin > x(N) THEN
        yout = 0
        GOTO label2:
    END IF

    klo = 1
    khi = N
label3:
    IF khi - klo > 1 THEN
        K% = (khi + klo) / 2
        IF x(K%) > xin THEN
            khi = K%
        ELSE
            klo = K%
        END IF
        GOTO label3:
    END IF
    h = x(khi) - x(klo)
    IF h = 0 THEN

```

```

        PRINT "Bad x input"
    END
END IF
yout = y(klo) + ((xin - x(klo)) / h) * (y(khi) - y(klo))
label2:
END SUB

DEFINT I-J
SUB Plot1 (de, kc%, NO(), h, T, P, EN)
'***** Linear Plot
*****
'***** Draw axes and tick marks
*****
Xmin = de                                'Determine range of values
for x-axis
Xmax = de * kc%
Max = NO(1)                              'Determine range of values
for y-axis
Min = 0
FOR I = 1 TO kc%
    IF NO(I) > Max THEN
        Max = NO(I)
    END IF
NEXT I
CLS 0                                    'Clear the screen
SCREEN 2                                'Set the display to 640 x 200
WINDOW
VIEW
LINE (64, 180)-(640, 180)              'Draw horizontal axis
FOR x = 0 TO 10                          'Draw tick marks on
horizontal axis
    LINE (x * 115 + 64, 176)-(x * 115 + 64, 182)
    LINE ((x * 115 / 2) + 64, 178)-((x * 115 / 2) + 64,
182)
NEXT x
LINE (64, 20)-(64, 180)                  'Draw vertical axis
FOR y = 0 TO 10                          'Draw tick marks on vertical
axis
    LINE (64, 180 - y * 16)-(68, 180 - y * 16)
NEXT y

'***** Label the Tick Marks
*****
LOCATE 1, 1
PRINT "Electron No. Density ";
PRINT "Number of bins ="; kc%; " de=";
PRINT USING "##.###"; de;
PRINT "eV h=";
PRINT USING "##.###-"; h
PRINT "    Linear Plot Temp = "; T; " Pressure = "; P /
110133#; "atm";

```

```

PRINT " E/W=";
PRINT USING "##.###-"; (EW) * 10000#;
PRINT " V cm^2"
YRange = Max - Min           'y-axis range
YIncr = YRange / 10          'incremental difference
between y-axis ticks
FOR y = 0 TO 10              'Label vertical axis tick
marks
    LOCATE y * 2 + 3, 1
    PRINT USING "##.####"; (Min + (10 - y) * YIncr)
NEXT y
'XMin = 0
XRange = Xmax - Xmin          'x-axis range
XIncr = XRange / 5            'incremental difference
between x-axis ticks
LOCATE 24, 5
FOR x = 0 TO 4                'Label horizontal axis tick
marks
    PRINT USING "##.####"; TAB((x) * 14 + 5); (Xmin + x *
XIncr);
NEXT x
PRINT USING "    ##.####"; (Xmin + 5 * XIncr);

'***** Plot the Data
*****
VIEW (64, 20)-(639, 180)     'Define view port to match axes
WINDOW (Xmin, Min)-(Xmax, Max) 'Scale pixel coordinates
FOR x = 1 TO kc% - 1
    x1 = Xmin + (XRange / (kc% - 1)) * (x - 1)
    y1 = NO(x)
    x2 = Xmin + (XRange / (kc% - 1)) * (x)
    y2 = NO(x + 1)
    LINE (x1, y1)-(x2, y2)
NEXT x
LOCATE 25, 1
PRINT "Send plot to printer ?";
L$ = INPUT$(1)
SELECT CASE L$
    CASE "y", "Y"
        LOCATE 25, 1: PRINT "
        CALL cgadump(HIRES)
END SELECT
SCREEN 0

'***** Log Plot
*****
'***** Draw axes and tick marks
*****
Xmin = de                     'Determine range of values
for

```

```

x-axis
Xmax = de * kc%
FOR I = 1 TO kc%
    NO(I) = .434294482# * LOG(NO(I)) 'Convert NO() to
common log
NEXT I
Max = NO(1) 'Determine range of values
for y-axis
Min = NO(1)
FOR I = 1 TO kc%
    IF NO(I) > Max THEN
        Max = NO(I)
    END IF
    IF NO(I) < Min THEN
        Min = NO(I)
    END IF
NEXT I
Min = INT(Min)
Max = INT(Max + 1)
CLS 0 'Clear the screen
SCREEN 2 'Set the display to 640 x 200
WINDOW
VIEW
LINE (64, 180)-(640, 180) 'Draw horizontal axis
FOR x = 0 TO 10 'Draw tick marks on
horizontal axis
    LINE (x * 115 + 64, 176)-(x * 115 + 64, 182)
    LINE ((x * 115 / 2) + 64, 178)-((x * 115 / 2) + 64,
182)
NEXT x
LINE (64, 20)-(64, 180) 'Draw vertical axis
FOR y = 0 TO 10 'Draw tick marks on vertical
axis
    LINE (64, 180 - y * 16)-(68, 180 - y * 16)
NEXT y

'***** Label the Tick Marks
*****
LOCATE 1, 1
PRINT "Electron No. Density ";
PRINT "Number of bins ="; kc%; " de=";
PRINT USING "##.###"; de;
PRINT "eV h=";
PRINT USING "##.###^---"; h
PRINT " Semi-Log Plot Temp = "; T; " Pressure = "; P /
110133#; "atm";
PRINT " E/N=";
PRINT USING "##.###^---"; (EN) * 10000#;
PRINT " V cm^-2"
YRange = Max - Min 'y-axis range
YIncr = YRange / 10 'incremental difference

```

```

between y-axis ticks
PRINT USING "###.##"; Max      'Label vertical axis tick
marks
FOR y = 1 TO 10
    LOCATE y * 2 + 3, 1
    PRINT USING "###.##"; (Min + (10 - y) * YIncr)
NEXT y
'XMin = 0
XRange = Xmax - Xmin           'x-axis range
XIncr = XRange / 5             'incremental difference
between x-axis ticks
LOCATE 24, 5
FOR x = 0 TO 4                 'Label horizontal axis tick
marks
    PRINT USING ".###^---"; TAB((x) * 14 + 5); (Xmin + x *
XIncr);
NEXT x
PRINT USING "      .###^---"; (Xmin + 5 * XIncr);

'***** Plot the Data
*****
VIEW (64, 20)-(639, 180)      'Define view port to match axes
WINDOW (Xmin, Min)-(Xmax, Max) 'Scale pixel coordinates
FOR x = 1 TO kc% - 1
    x1 = Xmin + (XRange / (kc% - 1)) * (x - 1)
    y1 = NO(x)
    x2 = Xmin + (XRange / (kc% - 1)) * (x)
    y2 = NO(x + 1)
    LINE (x1, y1)-(x2, y2)
NEXT x
LOCATE 25, 1
PRINT "Send plot to printer ?";
L$ = INPUT$(1)
SELECT CASE L$
    CASE "y", "Y"
        LOCATE 25, 1: PRINT "
";
        CALL cgadump(HIRES)
END SELECT
SCREEN 0
END SUB

SUB Plot2 (h1, h2, jChStep, ChTimeStep, tCav, jmax,
Yquant(), title$)
'***** Linear Plot
*****
'***** Draw axes and tick marks
*****
GOTO autocoords
stplot:
PRINT "Do you wish to specify the x & y ranges to be plotted

```

```

?"
L$ = INPUT$(1)
SELECT CASE L$
    CASE "y", "Y"
        GOTO mancoords
END SELECT
autocoords:      'sets x & y ranges to be plotted
automatically
Xmin = 0          'Determine range of values
for x-axis
IF ChTimeStep = 0 THEN
    Xmax = (h1 * tCav * 1E+09) * jmax
ELSE
    Xmax = (jChStep * h1 + (jmax - jChStep) * h2) * tCav *
1E+09
END IF
Max = Yquant(1)   'Determine range of
values
for y-axis
Min = 0
FOR I = 1 TO jmax
    IF Yquant(I) > Max THEN
        Max = Yquant(I)
    END IF
NEXT I
GOTO endcoords
mancoords:
INPUT "The minimum and maximum x values (ns) are: "; Xmin,
Xmax
PRINT "Do you wish to specify the y range to be plotted ?"
L$ = INPUT$(1)
SELECT CASE L$
    CASE "y", "Y"
        GOTO ycoords
END SELECT
Max = Yquant(1)   'Determine range of
values
for y-axis
Min = 0
FOR I = 1 TO jmax
    IF Yquant(I) > Max THEN
        Max = Yquant(I)
    END IF
NEXT I
GOTO endcoords
ycoords:
INPUT "The minimum and maximum y values are: "; Min, Max
endcoords:
CLS 0             'Clear the screen
SCREEN 2          'Set the display to 640 x 200
WINDOW

```

```

VIEW
LINE (64, 180)-(640, 180)      'Draw horizontal axis
FOR x = 0 TO 10                  'Draw tick marks on
horizontal axis
    LINE (x * 115 + 64, 176)-(x * 115 + 64, 182)
    LINE ((x * 115 / 2) + 64, 178)-((x * 115 / 2) + 64,
182)
NEXT x
LINE (64, 20)-(64, 180)        'Draw vertical axis
FOR y = 0 TO 10                  'Draw tick marks on vertical
axis
    LINE (64, 180 - y * 16)-(68, 180 - y * 16)
NEXT y

'***** Label the Tick Marks
*****
LOCATE 1, 1
PRINT title$; "    Linear Plot"
YRange = Max - Min              'y-axis range
YIncr = YRange / 10             'incremental difference
between y-axis ticks
FOR y = 0 TO 10                  'Label vertical axis tick
marks
    LOCATE y * 2 + 3, 1
    PRINT USING "#.##^----"; (Min + (10 - y) * YIncr)
NEXT y
'XMin = 0
XRange = Xmax - Xmin            'x-axis range
XIncr = XRange / 5              'incremental difference
between x-axis ticks
LOCATE 24, 5
FOR x = 0 TO 4                  'Label horizontal axis tick
marks
    PRINT USING ".###^----"; TAB((x) * 14 + 5); (Xmin + x *
XIncr);
NEXT x
PRINT USING "    .###^----"; (Xmin + 5 * XIncr);

'***** Plot the Data
*****
VIEW (64, 20)-(639, 180)      'Define view port to match axes
WINDOW (Xmin, Min)-(Xmax, Max) 'Scale pixel coordinates
IF ChTimeStep = 1 THEN GOTO varplot
FOR x = 1 TO jmax - 1
    y1 = Yquant(x)
    x1 = (h1 * tCav * 1E+09) * (x - 1)
    x2 = (h1 * tCav * 1E+09) * (x)
    y2 = Yquant(x + 1)
    LINE (x1, y1)-(x2, y2)
NEXT x

```

```

GOTO plotend
varplot:
FOR x = 1 TO jmax - 1
  IF x <= jChStep - 1 THEN
    x1 = (h1 * tCav * 1E+09) * (x - 1)
    x2 = (h1 * tCav * 1E+09) * (x)
    y1 = Yquant(x)
    y2 = Yquant(x + 1)
    LINE (x1, y1)-(x2, y2)
  END IF
  IF x = jChStep THEN
    x1 = (h1 * tCav * 1E+09) * (x - 1)
    x2 = (jChStep * h1 + (x - jChStep) * h2) * tCav *
1E+09
    y1 = Yquant(x)
    y2 = Yquant(x + 1)
    LINE (x1, y1)-(x2, y2)
  END IF
  IF x > jChStep THEN
    x1 = (jChStep * h1 + ((x - 1) - jChStep) * h2) *
tCav * 1E+09
    x2 = (jChStep * h1 + (x - jChStep) * h2) * tCav *
1E+09
    y1 = Yquant(x)
    y2 = Yquant(x + 1)
    LINE (x1, y1)-(x2, y2)
  END IF
NEXT x
plotend:
LOCATE 25, 1
PRINT "Send plot to printer ?";
L$ = INPUT$(1)
SELECT CASE L$
  CASE "y", "Y"
    LOCATE 25, 1: PRINT "
";
    CALL cgadump(HIRES)
END SELECT
LOCATE 25, 1: PRINT "
"
PRINT "Plot the Linear Plot again ?"
L$ = INPUT$(1)
SELECT CASE L$
  CASE "y", "Y"
    GOTO stplot
END SELECT

'***** Log Plot
*****
'***** Draw axes and tick marks
*****
FOR I = 1 TO jmax 'Convert Yquant() to common log
  Yquant(I) = .434294482# * LOG(Yquant(I))

```

```

NEXT I
GOTO autocords1
stplot1:
PRINT "Do you wish to specify the x & y ranges to be plotted
for
Log Plot ?"
L$ = INPUT$(1)
SELECT CASE L$
    CASE "y", "Y"
        GOTO mancoords1
END SELECT
autocords1:          'sets x & y ranges to be plotted
                    automatically
Xmin = 0              'Determine range of values
for x-axis
IF ChTimeStep = 0 THEN
    Xmax = (h1 * tCav * 1E+09) * jmax
ELSE
    Xmax = (jChStep * h1 + (jmax - jChStep) * h2) * tCav *
1E+09
END IF
Max = Yquant(1)      'Determine range of
values for y-axis
Min = Yquant(1)
FOR I = 1 TO jmax
    IF Yquant(I) > Max THEN
        Max = Yquant(I)
    END IF
    IF Yquant(I) < Min THEN
        Min = Yquant(I)
    END IF
NEXT I
Min = INT(Min)
Max = INT(Max + 1)
GOTO endcoords1
mancoords1:
INPUT "The minimum and maximum x values (ns) are: "; Xmin,
Xmax
PRINT "Do you wish to specify the y range to be plotted ?"
L$ = INPUT$(1)
SELECT CASE L$
    CASE "y", "Y"
        GOTO ycoords1
END SELECT
Max = Yquant(1)      'Determine range of
values for y-axis
Min = Yquant(1)
FOR I = 1 TO jmax
    IF Yquant(I) > Max THEN
        Max = Yquant(I)
    END IF

```

```

        IF Yquant(I) < Min THEN
            Min = Yquant(I)
        END IF
    NEXT I
    Min = INT(Min)
    Max = INT(Max + 1)
    GOTO endcoords1
ycoords1:
    INPUT "The minimum and maximum y values are:"; Min, Max
endcoords1:
    CLS 0          'Clear the screen
    SCREEN 2       'Set the display to 640 x 200
    WINDOW
    VIEW
    LINE (64, 180)-(640, 180)      'Draw horizontal axis
    FOR x = 0 TO 10                 'Draw tick marks on
horizontal axis
        LINE (x * 115 + 64, 176)-(x * 115 + 64, 182)
        LINE ((x * 115 / 2) + 64, 178)-((x * 115 / 2) + 64,
182)
    NEXT x
    LINE (64, 20)-(64, 180)        'Draw vertical axis
    FOR y = 0 TO 10                 'Draw tick marks on vertical
axis
        LINE (64, 180 - y * 16)-(68, 180 - y * 16)
    NEXT y

    '***** Label the Tick Marks
    *****
    LOCATE 1, 1
    PRINT title$; "    Log Plot"
    YRange = Max - Min              'y-axis range
    YIncr = YRange / 10             'incremental difference
between y-axis ticks
    LOCATE 3, 1
    PRINT USING "###.##"; Max
    FOR y = 1 TO 10                 'Label vertical axis tick
marks
        LOCATE y * 2 + 3, 1
        PRINT USING "###.##"; (Min + (10 - y) * YIncr)
    NEXT y
    'XMin = 0
    XRange = Xmax - Xmin            'x-axis range
    XIncr = XRange / 5              'incremental difference
between x-axis ticks
    LOCATE 24, 5
    FOR x = 0 TO 4                  'Label horizontal axis tick
marks
        PRINT USING ".###^---"; TAB((x) * 14 + 5); (Xmin + x *
XIncr);
    NEXT x

```

```

PRINT USING "      .##^----"; (Xmin + 5 * XIncr);

'***** Plot the Data
*****
VIEW (64, 20)-(639, 180)      'Define view port to match axes
WINDOW (Xmin, Min)-(Xmax, Max) 'Scale pixel coordinates
IF ChTimeStep = 1 THEN GOTO varplot1
FOR x = 1 TO jmax - 1
    y1 = Yquant(x)
    x1 = (h1 * tCav * 1E+09) * (x - 1)
    x2 = (h1 * tCav * 1E+09) * (x)
    y2 = Yquant(x + 1)
    LINE (x1, y1)-(x2, y2)
NEXT x
GOTO plotendi
varplot1:
FOR x = 1 TO jmax - 1
    IF x <= jChStep - 1 THEN
        x1 = (h1 * tCav * 1E+09) * (x - 1)
        x2 = (h1 * tCav * 1E+09) * (x)
        y1 = Yquant(x)
        y2 = Yquant(x + 1)
        LINE (x1, y1)-(x2, y2)
    END IF
    IF x = jChStep THEN
        x1 = (h1 * tCav * 1E+09) * (x - 1)
        x2 = (jChStep * h1 + (x - jChStep) * h2) * tCav *
1E+09
        y1 = Yquant(x)
        y2 = Yquant(x + 1)
        LINE (x1, y1)-(x2, y2)
    END IF
    IF x > jChStep THEN
        x1 = (jChStep * h1 + ((x - 1) - jChStep) * h2) *
tCav * 1E+09
        x2 = (jChStep * h1 + (x - jChStep) * h2) * tCav *
1E+09
        y1 = Yquant(x)
        y2 = Yquant(x + 1)
        LINE (x1, y1)-(x2, y2)
    END IF
NEXT x
plotendi:
LOCATE 25, 1
PRINT "Send plot to printer ?";
L$ = INPUT$(1)
SELECT CASE L$
    CASE "y", "Y"
        LOCATE 25, 1: PRINT "
";
        CALL cgadump(HIRES)

```

```
END SELECT
LOCATE 25, 1: PRINT "
PRINT "Plot Log Plot again ?"
L$ = INPUT$(1)
SELECT CASE L$
    CASE "y", "Y"
        GOTO stplot1
END SELECT
SCREEN 0
END SUB
```

"

## Appendix D. *Data Files*

### *D.1 Nitrogen*

8

34,N2 V=1,.29

.29,0

.3,1E-19

.33,2E-19

.4,3E-19

.75,5E-19

.9,6.5E-19

1,8E-19

1.1,1E-19

1.165,1.2E-18

1.2,1.37E-18

1.218,1.5E-18

1.4,6.75E-18

1.5,9.5E-18

1.6,1.22E-17

1.65,1.39E-17

1.7,1.6E-17

1.8,3.3E-17

1.9,1.52E-16

2,1.32E-16

2.1,4.6E-17

2.2,1.63E-16

2.3,1.23E-16

2.4,4.6E-17

2.5,8.6E-17

2.6,1.04E-16

2.7,2.7E-17

2.8,4.2E-17

2.9,4.27E-17

3.0,4.3E-17

3.1,5.8E-17

3.2,3.8E-17

3.3,2.9E-17

3.6,2.9E-17

5.0,0

18,N2 V=2,.59

1.7,0

1.8,9E-18

1.9,4E-17

2.0,1.52E-16

2.1,1.4E-16

2.2,6.2E-17

2.3,6E-17

2.4,1.39E-16

2.5,1.14E-16  
 2.6,3.1E-17  
 2.7,4.9E-17  
 2.8,5.1E-17  
 2.9,1.8E-17  
 3.0,2.4E-17  
 3.1,1.5E-17  
 3.2,1.1E-17  
 3.3,7E-18  
 3.4,0  
 17,N2 V=3,.89  
 1.8,0  
 1.9,1.8E-17  
 2.0,7.5E-17  
 2.1,1.41E-16  
 2.2,1.69E-16  
 2.3,8.5E-17  
 2.4,2.9E-17  
 2.5,7.7E-17  
 2.6,1.17E-16  
 2.7,6.4E-17  
 2.8,2.6E-17  
 2.9,4E-17  
 3.0,4E-17  
 3.1,1.6E-17  
 3.2,1.6E-17  
 3.3,1.6E-17  
 3.4,0  
 16,N2 V=4,1.17  
 1.9,0  
 2.0,1.6E-17  
 2.1,4.6E-17  
 2.2,1.1E-16  
 2.3,1.3E-16  
 2.4,7.1E-17  
 2.5,2E-17  
 2.6,3.1E-17  
 2.7,6E-17  
 2.8,4.9E-17  
 2.9,1.8E-17  
 3.0,1.6E-17  
 3.1,1.6E-17  
 3.2,1.1E-17  
 3.3,7E-18  
 3.4,0  
 15,N2 V=5,1.47  
 2.0,0  
 2.1,2E-17  
 2.2,4.6E-17  
 2.3,7.7E-17  
 2.4,1.04E-16

2.5,1.01E-16  
 2.6,5.1E-17  
 2.7,2.7E-17  
 2.8,3.7E-17  
 2.9,6.2E-17  
 3.0,4.2E-17  
 3.1,2.7E-17  
 3.2,3.5E-17  
 3.3,3.1E-17  
 3.4,0  
 13,N2 V=6,1.76  
 2.2,0  
 2.3,1.1E-17  
 2.4,3.7E-17  
 2.5,6E-17  
 2.6,6E-17  
 2.7,3.7E-17  
 2.8,1.5E-17  
 2.9,9E-18  
 3.0,1.6E-17  
 3.1,1.8E-17  
 3.2,7E-18  
 3.3,5E-18  
 3.4,0  
 12,N2 V=7,2.06  
 2.3,0  
 2.4,7E-18  
 2.5,1.8E-17  
 2.6,2.9E-17  
 2.7,4.4E-17  
 2.8,3.3E-17  
 2.9,1.8E-17  
 3.0,5E-18  
 3.1,7E-18  
 3.2,1.6E-17  
 3.3,7E-18  
 3.4,0  
 8,N2 V=8,2.35  
 2.5,0  
 2.6,7E-18  
 2.7,1.1E-17  
 2.8,1.8E-17  
 2.9,2.4E-17  
 3.0,1.5E-17  
 3.1,7E-18  
 3.2,0  
 50,N2  
 0,1E-16  
 1E-4,1E-16  
 4E-4,1.2E-16  
 9E-4,1.33E-16

1.6E-3, 1.43E-16  
2.5E-3, 1.56E-16  
3.6E-3, 1.69E-16  
4.9E-3, 1.8E-16  
6.4E-3, 1.94E-16  
8.1E-3, 2.05E-16  
1.03E-2, 2.2E-16  
1.44E-2, 2.49E-16  
2.21E-2, 2.94E-16  
3.24E-2, 3.5E-16  
4.0E-2, 3.86E-16  
4.84E-2, 4.24E-16  
6.51E-2, 4.9E-16  
7.86E-2, 5.33E-16  
.103, 6.04E-16  
.1156, 6.31E-16  
.1502, 7.12E-16  
.226, 8.22E-16  
.332, 9.34E-16  
.445, 9.95E-16  
1.0, 9.98E-16  
1.1, 1.014E-15  
1.2, 1.051E-15  
1.3, 1.18E-15  
1.4, 1.145E-15  
1.5, 1.196E-15  
1.6, 1.29E-15  
1.7, 1.343E-15  
1.8, 1.695E-15  
1.9, 1.983E-15  
2.0, 2.401E-15  
2.2, 2.876E-15  
2.6, 2.988E-15  
2.8, 2.801E-15  
3.0, 2.163E-15  
3.3, 1.719E-15  
3.6, 1.466E-15  
4.0, 1.262E-15  
4.5, 1.152E-15  
6.0, 1.03E-15  
10, 8.51E-16  
15, 1.1E-15  
20, 1.2E-15  
25, 1.17E-15  
35, 1.05E-15  
40, 1.01E-15

D.2 CO2

9

36,CO2 v=1,.0832

0,0

.0827,0

.0844,8.5E-17

.0862,1.16E-16

.0932,1.85E-16

.1035,2.3E-16

.1208,2.6E-16

.1382,2.68E-16

.1726,2.62E-16

.207,2.48E-16

.275,2.18E-16

.345,1.93E-16

.5,1.45E-16

.7,1.1E-16

.9,8E-17

1.1,6.2E-17

1.4,4.6E-17

1.6,4.2E-17

1.8,4.4E-17

2.3,7E-17

2.6,9.3E-17

3.0,1.34E-16

3.2,1.58E-16

3.4,1.75E-16

3.6,1.8E-16

3.8,1.79E-16

4.0,1.7E-16

4.2,1.52E-16

4.6,1.05E-16

5.1,5.7E-17

5.5,5.1E-17

6,5E-17

7,4.8E-17

8,4.5E-17

10,2E-17

20,0

27,CO2 v=2,.167

0,0

.167,0

.2,6E-17

.22,7.6E-17

.25,8E-17

.3,7.8E-17

.5,6.4E-17

.7,5.3E-17

1,4.4E-17

1.25,4.4E-17  
 1.5,4.4E-17  
 2.0,5.3E-17  
 2.5,8.4E-17  
 3.0,1.28E-16  
 3.2,1.57E-16  
 3.4,1.77E-16  
 3.55,1.78E-16  
 3.7,1.75E-16  
 3.9,1.6E-16  
 4.1,1.28E-16  
 4.5,8.8E-17  
 4.9,3.9E-17  
 5.2,3.3E-17  
 6.0,2.7E-17  
 8.0,2.5E-17  
 10,2.1E-17  
 20,0  
 18,CO2 v=3,.291  
 .29,0  
 .297,4E-17  
 .303,5E-17  
 .327,6.2E-17  
 .364,7.1E-17  
 .425,7.7E-17  
 .485,8.4E-17  
 .607,9.2E-17  
 .728,9.7E-17  
 .969,9.9E-17  
 1.21,9.5E-17  
 2.43,6.6E-17  
 4.85,4.4E-17  
 9.68,2.6E-17  
 18.2,1.5E-17  
 36.4,9.2E-18  
 58.2,5.8E-18  
 100,0  
 8,CO2 v=4,.339  
 1.5,0  
 1.95,7E-18  
 2.5,2E-17  
 3.0,4.1E-17  
 3.56,6.6E-17  
 4.1,3.4E-17  
 4.5,1.55E-17  
 5.06,0  
 7,CO2 v=5,.252  
 2.5,0  
 3.0,1.05E-17  
 3.56,5.4E-17  
 4.1,3.4E-17

4.5,1.6E-17  
 5.06,4.4E-18  
 6.0,0  
 5,C02 v=6,.422  
 2.5,0  
 3.0,1.05E-17  
 3.56,2.25E-17  
 4.1,1E-17  
 4.5,0  
 5,C02 v=7,.505  
 2.5,0  
 3.0,1.56E-17  
 3.56,3.3E-17  
 4.1,1.56E-17  
 4.5,0  
 5,C02 v=8,2.5  
 2.5,0  
 3.0,1.8E-17  
 3.6,2.5E-17  
 4.1,1.8E-17  
 4.5,0  
 10,C02 v=9,3.85  
 3.65,0  
 4.3,1.4E-19  
 4.5,1.4E-19  
 5.1,0  
 6.6,0  
 7.2,7E-20  
 8.2,4.5E-19  
 8.4,4.2E-19  
 8.9,1E-19  
 9.7,0  
 44,C02  
 0.0,6E-14  
 .001,5.4E-14  
 .002,3.8E-14  
 .004,2.7E-14  
 .007,2E-14  
 .01,1.7E-14  
 .02,1.2E-14  
 .04,8.5E-15  
 .07,6.4E-15  
 .1,5.2E-15  
 .15,4E-15  
 .2,3.15E-15  
 .25,2.5E-15  
 .3,2E-15  
 .35,1.65E-15  
 .42,1.3E-15  
 .5,1.08E-15  
 .6,8.8E-16

.7,1.1E-16  
 .85,6.3E-16  
 1.0,5.6E-16  
 1.25,5.1E-16  
 1.5,5E-16  
 1.8,5E-16  
 2.2,5.4E-16  
 2.5,6.5E-16  
 2.8,7.6E-16  
 3.2,1.03E-15  
 3.6,1.42E-15  
 4.0,1.52E-15  
 4.5,1.48E-15  
 4.9,1.32E-15  
 5.2,1.2E-15  
 5.6,1.08E-15  
 6.4,9.8E-16  
 8.0,1.08E-15  
 10,1.21E-15  
 14,1.41E-15  
 18,1.56E-15  
 28,1.62E-15  
 40,1.46E-15  
 52,1.27E-15  
 75,9.6E-16  
 100,8E-16

### *D.3 Helium*

38,He  
 0.0,5.18E-16  
 .008,5.18E-16  
 .009,5.198E-16  
 .01,5.21E-16  
 .013,5.26E-16  
 .017,5.31E-16  
 .02,5.35E-16  
 .025,5.41E-16  
 .03,5.46E-16  
 .04,5.54E-16  
 .05,5.62E-16  
 .06,5.69E-16  
 .07,5.74E-16  
 .08,5.79E-17  
 .09,5.83E-16  
 .1,5.86E-16  
 .12,5.94E-16  
 .15,6.04E-16

.18,6.12E-16  
.2,6.16E-16  
.25,6.27E-16  
.3,6.35E-16  
.4,6.49E-16  
.5,6.59E-16  
.6,6.66E-16  
.7,6.73E-16  
.8,6.77E-16  
.9,6.82E-16  
1,6.85E-16  
1.2,6.91E-16  
1.5,6.96E-16  
1.8,6.98E-16  
2.0,6.99E-16  
2.5,6.96E-16  
3.0,6.89E-16  
4.0,6.6E-16  
5.0,6.26E-16  
6.0,6.01E-16

## Appendix E. *Methods of Numerical Solution*

### E.1 *Gauss-Jordan*

The Gauss-Jordan method calculates the inverse of a matrix through a series of steps called normalization and reduction (11:193). The first step in this process is to form an augmented matrix using the  $\tilde{A}$  matrix which is to be inverted, and matrix  $\tilde{I}$ , the identity matrix.

$$[\tilde{A}\tilde{I}] = \begin{pmatrix} a_{11} & a_{12} & a_{13} & 1 & 0 & 0 \\ a_{21} & a_{22} & a_{23} & 0 & 1 & 0 \\ a_{31} & a_{32} & a_{33} & 0 & 0 & 1 \end{pmatrix} \quad (51)$$

The second step is to normalize all the elements of the first row by its diagonal element. The normalized first row is now multiplied by an appropriately chosen constant such that when subtracted from the second row the result will reduce at least one of the second row's elements to zero. This process of row reduction using the normalized first row is performed on each row (excluding the normalized row). When reduction using the first row is complete, the second row is normalized by its diagonal element, and reduction of all the other rows performed. This process is repeated until the last row has been normalized. Once complete, the original matrix  $\tilde{A}$  will have been reduced to the identity matrix, and the  $\tilde{I}$  matrix will have been replaced by the inverse of  $\tilde{A}$ . This method can be used to solve for the inverse of  $(I - hC)$  in equation (42). An initial guess distribution is then used to iteratively solve for the electron number density distribution. The initial guess distribution used was a straight line distribution (constant value equal to one). Fortunately, this inverse need only be calculated once for a particular solution. The iterative process of equation (42) is repeated until the desired convergence criteria ( $10^{-5}$ ) is met. The algorithm used for the Gauss-Jordan solution was taken from reference 11.

## E.2 L-U Decomposition

L-U decomposition is another method which attempts to solve equation (42) by calculation of the inverse of  $(I - hC)$ . This technique is based upon a decomposition of the original matrix  $\vec{A}$  into a lower triangular part and an upper triangular part. These are denoted  $\vec{L}$  and  $\vec{U}$ , such that

$$\vec{L} \cdot \vec{U} = \vec{A} \quad (52)$$

and we wish to solve the linear equation

$$\vec{A} \cdot \vec{x} = \vec{b} \quad (53)$$

such that

$$\vec{A} \cdot \vec{x} = (\vec{L} \cdot \vec{U}) \cdot \vec{x} = \vec{L} \cdot (\vec{U} \cdot \vec{x}) = \vec{b} \quad (54)$$

We accomplish this by first solving for the vector  $\vec{y}$  where

$$\vec{L} \cdot \vec{y} = \vec{b} \quad (55)$$

and then solving for  $\vec{x}$  by

$$\vec{U} \cdot \vec{x} = \vec{y} \quad (56)$$

Once we have L-U decomposed a particular matrix  $\vec{A}$ , we can solve for its inverse one column at a time, by successively letting the  $\vec{b}$  vector be a column of the identity matrix. The resulting columns when put together will form the inverse of the  $\vec{A}$  matrix. The program for this procedure comes from Press (10:31). This inverted  $(I - hC)$  matrix was then used iteratively in equation (42) with the initial guess being a straight line (constant distribution).

### E.3 Gauss-Seidel

Referring to equation (40), we see that for the steady state case

$$\sum_l C_{kl} n_l = 0 \quad (57)$$

Assuming once again that the elements of the C matrix are constants, we are left with solving a set of linear equations. Iterative algorithms can offer a fast, accurate solution in cases where the C matrix is large (13:382). One such technique is the Gauss-Seidel algorithm. For solving equation (49), this algorithm takes the form (9:373)

$$n_k^{(i+1)} = -\frac{1}{C_{kl}} \left( \sum_{l=1}^{k-1} C_{kl} n_l^{(i)} + \sum_{l=k+1}^K C_{kl} n_l^{(i+1)} \right) \quad (58)$$

for  $k = K$  to  $k = 1$ . This is an improvement over a technique such as the Jacobi method because as soon as they are available, updated values are used to calculate in the second term on the right to calculate estimates of the same iterative order. The rate of convergence may be improved through the use of Aitken's acceleration technique (9:373). The formula for this algorithm is

$$n_k^{(i+1)} = n_k^{(i-2)} - \frac{(n_k^{(i-2)} - n_k^{(i-1)})^2}{n_k^{(i-2)} - 2n_k^{(i-1)} + n_k^{(i)}} \quad (59)$$

A potential draw back with the Gauss-Seidel method lies with the initial guess that must be supplied. If the initial guess is far off, convergence may be very time consuming (even with the use of an accelerator). In order to limit this problem, the input initial guess used came from 20, 30 and 50 bin calculations using the L-U decomposition method. An initial guess constant value distribution was also tried.

#### E.4 Successive Over Relaxation (SOR)

Let us denote  $\mathbf{x}$  as an approximation to a linear system  $A\mathbf{x} = \vec{b}$ . We will define a residual vector with respect to this system as  $\vec{r} = \vec{b} - A\mathbf{x}$ . Using the notation of the residual vector, the Gauss-Seidel method may be rewritten as (13:391)

$$x_i^{(k)} = x_i^{(k-1)} + \frac{r_{ii}^{(k)}}{a_{ii}} \quad (60)$$

This in turn may be modified such that

$$x_i^{(k)} = x_i^{(k-1)} + \omega \frac{r_{ii}^{(k)}}{a_{ii}} \quad (61)$$

Certain choices of  $\omega$  may lead to faster convergence for equation (53). Methods of solution based upon equation (53) are called relaxation methods. Under-relaxation is the region where  $0 < \omega < 1$ . Over-relaxation is the region where  $\omega > 1$ . Both of these methods are abbreviated as Successive Over-Relaxation (SOR). Using this approach, we rewrite equation (50) as (13:391)

$$x_i^{(k)} = (1 - \omega)x_i^{(k-1)} + \frac{\omega}{a_{ii}} \left[ -\sum_{j=1}^{i-1} a_{ij}x_j^{(k)} - \sum_{j=i+1}^n a_{ij}x_j^{(k-1)} \right] \quad (62)$$

for  $i = 1$  to  $K$ . The advantage of this method over Gauss-Seidel is that it may provide accelerated convergence. According to the theorem of Kahan (13:392), we must have  $0 < \omega < 2$  if  $a_{ii} \neq 0$  for each  $i = 1, 2, \dots, K$ . Otherwise the solution will not be convergent. As in the case of the Gauss-Seidel method, L-U decomposition and a straight line distribution were used to seed the initial guess vectors.

### E.5 Speed and Accuracy Comparison

To judge how well a particular method worked, a normalized ( $\sum n(\epsilon)\Delta\epsilon = 1$ ) electron number density distribution was calculated using the above methods. Double precision was used in all calculations. The computer used was an Apple II GS with an Applied Engineering PC Transporter, math coprocessor and 640K of ram installed. The cross section data used was for nitrogen (14:62-67). From this distribution function, the drift velocity, relative convergence of the distribution and the energy balance were calculated. To determine the energy balance, the rate at which the electrons gained energy from the field and the rates at which the electrons lost energy through elastic and inelastic collisions were all calculated. From equation (39), (56) and (57) we get

$$\begin{aligned}\dot{E}_g &= \sum_k (\bar{a}_k - \bar{b}_k) n_k \Delta\epsilon \\ \dot{E}_{el} &= - \sum_k [(a_k - \bar{a}_k) - (b_k - \bar{b}_k)] n_k \Delta\epsilon \\ \dot{E}_{inel} &= - \sum_k [N_s (R_{js,k+m_{js}} n_{k+m_{js}} m_{js} - \Delta_{js} R'_{js} n_k - m_{js})] \Delta\epsilon \\ \text{Energy Balance} &= \frac{E_g - E_{el} - E_{inel}}{E_g}\end{aligned}\quad (63)$$

A convergence value for each element of the distribution was calculated according to

$|n_k^m - n_k^{m-1}|/n_k^{m-1}$ . The convergence criteria used was  $10^{-5}$ .

The number of iterations and run time were also recorded. The calculated drift velocity was compared with that for an experimentally reported value (15:627-628) with  $E/N = 5 \times 10^{-17}$  V cm<sup>2</sup> at a temperature of 293 K. The results are reported in Table 12.

In Table 12, the best results obtained by each method are reported. The two methods used to seed the initial guess for the last three methods were L-U decomposition and a straight line distribution. Runs were made using 20, 30 and 50 bins to generate the initial guess distribution by L-U decomposition. The best results were obtained using fifty bins, although this increased run time. L-U decomposition providing the initial guess always gave better speed and equal or better

Table 12. Comparison of Algorithms					
Method	vd***	Converg	Balance	Iter	Run Time
Experimental	1.095				
L-U decomp	1.083	$1.28 \times 10^{-7}$	$4.50 \times 10^{-4}$	4	9.5
Gauss-Jordan	1.083	$2.76 \times 10^{-6}$	$4.50 \times 10^{-4}$	4	18
Gauss-Seidel*	1.079	$9.98 \times 10^{-6}$	$4.51 \times 10^{-3}$	1188	90
Gauss-Seidel**	—	D	—	—	—
SOR	1.081	$9.86 \times 10^{-6}$	$1.48 \times 10^{-4}$	112	10.5

\* - no acceleration technique is applied

\*\* - Aitken's acceleration technique is applied

\*\*\* - velocity is  $\times 10^6$  cm/s

D - did not converge

All run times are in minutes, convergence criteria was  $10^{-5}$

accuracy than the straight line distribution.

Gauss-Seidel with Aitken's acceleration did not converge to a stable solution. An oscillatory component was present in the convergence calculations, and this prevented the calculation from establishing a sufficiently strong overall downward trend. This same result occurred regardless of the initial guess used.

SOR also exhibited an oscillatory behavior in the convergence calculation. However, this behavior was weak enough so that the overall trend in convergence was downward. Eventually, the convergence criteria was met. The best SOR results were obtained with a 50 bin L-U decomposition initial guess and  $\omega = 1.9$ .

L-U decomposition and Gauss-Jordan provided results of comparable accuracy. Run time for Gauss-Jordan is however nearly twice that of L-U decomposition. This factor of two is due to the greater number of computations required of the Gauss-Jordan method. About eighty five percent of run time was spent on matrix inversion for both L-U decomposition and Gauss-Jordan. Experience has shown that an energy balance of  $10^{-4}$  or smaller (meaning a better energy balance) should be obtainable for a good choice of energy range, bin width, number of bins and time step. An energy balance of greater than  $10^{-3}$  has usually meant the presence of an instability which might be seen by examination of the linear and log plots of the number density. Of all the methods tried, L-U decomposition provided the best accuracy and run time.

## Bibliography

1. Patel, C.K.N. "High-Power Carbon Dioxide Lasers," *Scientific American*, 219: 22-33 (August 1968).
2. Smith, Kenneth and R.M. Thompson. *Computer Modeling of Gas Lasers*. New York: Plenum Press, 1978.
3. Witteman, W.J. *The CO<sub>2</sub> Laser*. Berlin: Springer-Verlag, 1987.
4. Holstein, T. "Energy Distribution in High Frequency Gas Discharges," *Physical Review*, 70: 367-371 (September 1946).
5. Lacina, William B. "Theoretical Modeling of Molecular and Electron Kinetic Processes." Northrop Research and Technology Center Report NRTC-79-7R. Palos Verdes Peninsula CA, January 1979.
6. Book, David L. *NRL Plasma Formulary*. NRL Publication 0084-4040, Washington DC, 1987.
7. Rockwood, Stephen D. "Elastic and Inelastic Cross Sections for Electron-Hg Scattering from Hg Transport Data," *Physical Review A*, 8: 2348-2358 (November 1973).
8. Rockwood, S.D. and Greene, A.E. "Numerical Solutions of the Boltzmann Transport Equation," *Computer Physics Communications*, 19: 377-393 (July-August 1980).
9. Thompson, R.M. and others. "Boltz: A Code to Solve the Transport Equation for Electron Distributions and then Calculate Transport Coefficients and Vibrational Excitation Rates in Gases with Applied Fields," *Computer Physics Communications*, 11: 369-383 (June-August 1976).
10. Press, William H. and others. *Numerical Recipes*. Cambridge: Cambridge University Press, 1986.
11. Constantinides, Alkis. *Applied Numerical Methods with Personal Computers*. New York: McGraw-Hill, Inc., 1987.
12. Marion, Jerry B. and Mark A. Heald. *Classical Electromagnetic Radiation*. Orlando: Academic Press, 1980.
13. Burden, Richard L. and others. *Numerical Analysis*. Boston: Prindle, Weber and Schmidt, 1981.
14. Kieffer, L.J. "A Compilation of Electron Collision Cross Section Data for Modeling Gas Discharge Lasers," *JILA Information Center Report 13*. Boulder: University of Colorado, 30 September 1973.
15. Huxley, L.G.H. and R.W. Crompton. *The Diffusion and Drift of Electrons in Gases*. New York: John Wiley and Sons, 1974.
16. Cooper, James W. *Microsoft QuickBasic for Scientists*. New York: John Wiley and Sons, 1988.
17. Shkarofsky, I.P. and others. *The Particle Kinetics of Plasmas*. Reading MA: Addison-Wesley Publishing Company, 1966.
18. Elliot C.J. and others. *Electron Transport Coefficients and Vibrational Excitation Rates for Electrically Excited CO<sub>2</sub> Gas Lasers*. Informal Report LA-5562-MS. Los Alamos: Los Alamos Scientific Laboratory, 1974.
19. Verdeyen, Joseph T. *Laser Electronics*. Englewood Cliffs NJ: Prentice-Hall, Inc., 1981.

20. Nickel, George H. *Transport of Spectral Line Radiation in Cylindrical Plasmas*. Los Alamos Report LA-UR-86-580. Los Alamos: Los Alamos National Laboratory.
21. Davies, A.R. and others. "Calculations on Output Pulse Shapes, Gain Pulse Profiles, and Gain Limitation in the CO<sub>2</sub> TEA Laser," *Journal of Applied Physics*, **47**: 2037-2043 (May 1976).
22. Denes, L.J. and J.J. Lowke. "V-I Characteristics of Pulsed CO<sub>2</sub> Laser Discharges," *Applied Physics Letters*, **23**: 130-132 (August 1973).

### *Vita*

Captain David A. Honey was born on 2 January 1954 in Wilmington, Delaware. He graduated from high school in 1972 and entered the Rochester Institute of Technology, from which he received the Bachelor of Science in Photographic Science in June 1976. He then attended the University of Arizona and received a Master of Science in Optical Science in 1978. Upon graduation, he entered the USAF and was commissioned through OTS. He attended pilot training and graduated in January 1980. He served as a B-52 pilot and then as a FB-111A pilot and instructor/flight examiner, until entering the School of Engineering, Air Force Institute of Technology, in June 1988.

Permanent address: 31 North Cliffe Drive  
Wilmington, Delaware 19809

# REPORT DOCUMENTATION PAGE

Form Approved  
OMB No. 0704-0188

1a. REPORT SECURITY CLASSIFICATION UNCLASSIFIED			1b. RESTRICTIVE MARKINGS	
2a. SECURITY CLASSIFICATION AUTHORITY			3. DISTRIBUTION / AVAILABILITY OF REPORT Approved for public release; distribution unlimited	
2b. DECLASSIFICATION / DOWNGRADING SCHEDULE				
4. PERFORMING ORGANIZATION REPORT NUMBER(S) AFIT, GEP, ENP/89D-5			5. MONITORING ORGANIZATION REPORT NUMBER(S)	
6a. NAME OF PERFORMING ORGANIZATION School of Engineering		6b. OFFICE SYMBOL (if applicable) AFIT, ENP	7a. NAME OF MONITORING ORGANIZATION	
6c. ADDRESS (City, State, and ZIP Code) Air Force Institute of Technology Wright-Patterson AFB OH 45433-5583			7b. ADDRESS (City, State, and ZIP Code)	
8a. NAME OF FUNDING / SPONSORING ORGANIZATION WRDC		8b. OFFICE SYMBOL (if applicable) AARI	9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER	
8c. ADDRESS (City, State, and ZIP Code) WRIGHT-PATTERSON AFB OH 45433			10. SOURCE OF FUNDING NUMBERS	
			PROGRAM ELEMENT NO.	PROJECT NO.
			TASK NO.	WORK UNIT ACCESSION NO.
11. TITLE (Include Security Classification) A NUMERICAL SOLUTION TO THE BOLTZMANN EQUATION FOR USE IN CALCULATING PUMPING RATES IN A CO2 DISCHARGE LASER				
12. PERSONAL AUTHOR(S) David A. Honey, B.S., M.S., Capt, USAF				
13a. TYPE OF REPORT MS Thesis		13b. TIME COVERED FROM _____ TO _____	14. DATE OF REPORT (Year, Month, Day) 1989 December	15. PAGE COUNT 130
16. SUPPLEMENTARY NOTATION				
17. COSATI CODES			18. SUBJECT TERMS (Continue on reverse if necessary and identify by block number)  CO2 Laser Boltzmann Equation	
FIELD	GROUP	SUB-GROUP		
20	09			
09	03			
19. ABSTRACT (Continue on reverse if necessary and identify by block number)  Thesis Advisor: Maj David Stone Associate Professor Department of Engineering Physics				
20. DISTRIBUTION / AVAILABILITY OF ABSTRACT <input checked="" type="checkbox"/> UNCLASSIFIED/UNLIMITED <input type="checkbox"/> SAME AS RPT. <input type="checkbox"/> DTIC USERS			21. ABSTRACT SECURITY CLASSIFICATION UNCLASSIFIED	
22a. NAME OF RESPONSIBLE INDIVIDUAL Maj David Stone, Assoc. Professor			22b. TELEPHONE (Include Area Code) (513)255-3030	22c. OFFICE SYMBOL ENP

UNCLASSIFIED

The collisional Boltzmann equation was solved numerically to obtain excitation rates for use in a CO2 laser design program. The program was written in Microsoft QuickBasic for use on the IBM Personal Computer or equivalent.

Program validation involved comparisons of computed transport coefficients with experimental data and previous theoretical work. Four different numerical algorithms were evaluated in terms of accuracy and efficiency. L-U decomposition was identified as the preferred approach. The calculated transport coefficients were found to agree with empirical data within one to five percent.

The program was integrated into a CO2 laser design program. Studies were then performed to evaluate the effects on predicted laser output power and energy density as parameters affecting electron kinetics were changed. Plotting routines were written for both programs.

UNCLASSIFIED